

Angular Line of Business Apps Hands On

Deborah Kurata

Jerry Kurata

Consultants | Developers | Mentors

deborahk@insteptech.com
jerryk@insteptech.com

Deborah Kurata

- Independent Consultant | Developer | Mentor
 - Web (Angular), .NET
- Pluralsight Author
 - AngularJS Line of Business Applications
 - Angular Front to Back with Web API
 - Object-Oriented Programming Fundamentals in C#
- Microsoft MVP

Jerry Kurata

- Independent Consultant | Developer | Mentor
 - Web (Angular), .NET
 - Internet of Things (IoT)
 - Big Data
- Speaker

Rate Yourself on **Angular**

- New to Angular - **no** proficiency
- Just starting out - **limited** proficiency
- Doing it, not fully understanding it - **working** proficiency
- Been there, done that, can help others - **full** proficiency

Overview



Basics

Navigation and Routing

Data Access

Forms and Validation

Hands On Labs

- After the topic has been presented
- Starter files provided
 - Minimize excess typing
- Detailed instructions provided
 - With the "answer" to use as needed
- Extra Tasks
 - If time permits

BASICS

Overview - Basics



Introduction to Angular

Angular and HTML: Building a View

Data Binding

Angular Code: Building the Module & Controller

Bootstrap: Styling the View

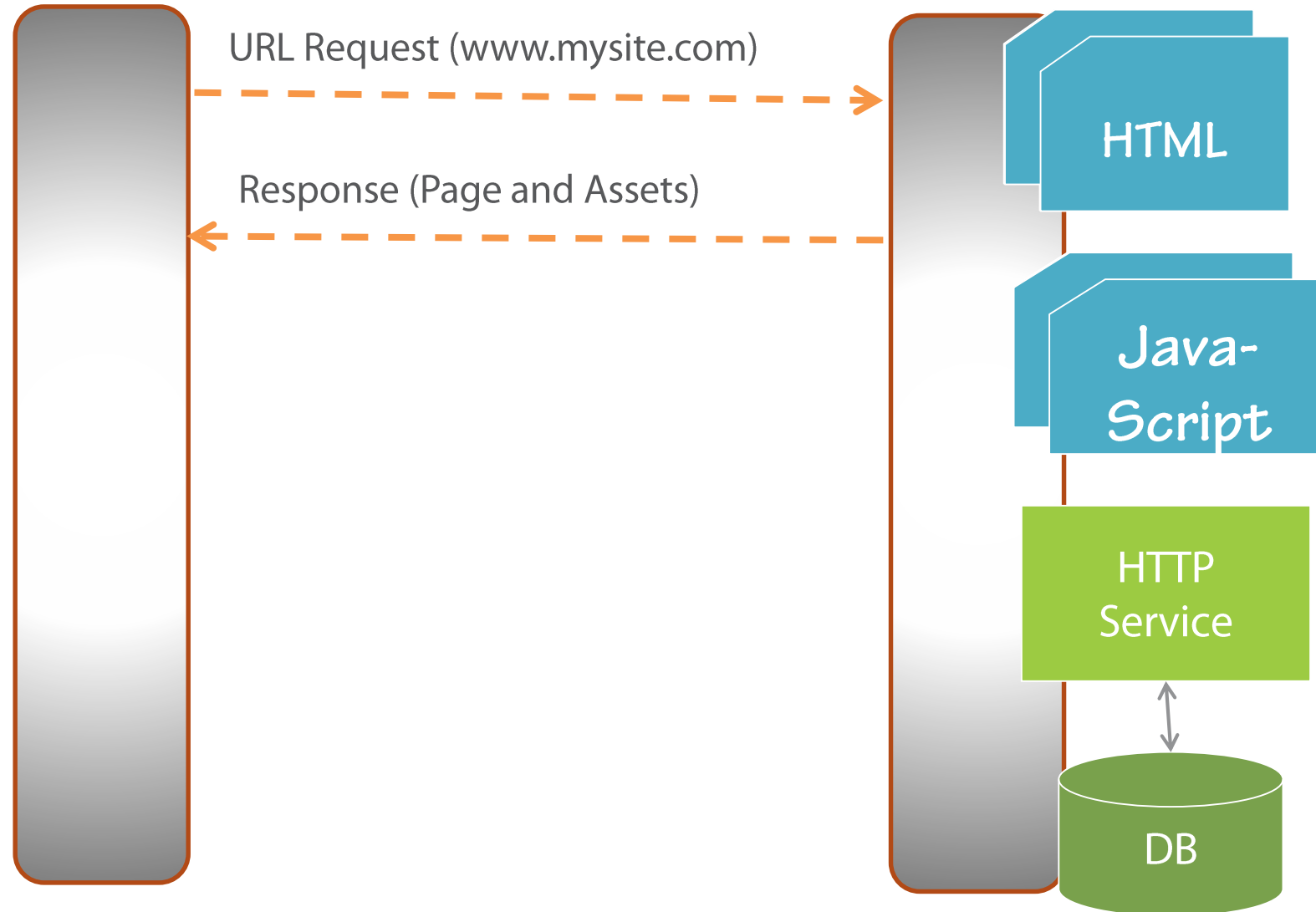


- A client-side JavaScript framework for building interactive Web applications
- Brings simple/clean back to complex Web apps
- Originally developed by Google
- Now open source: <https://angularjs.org/>



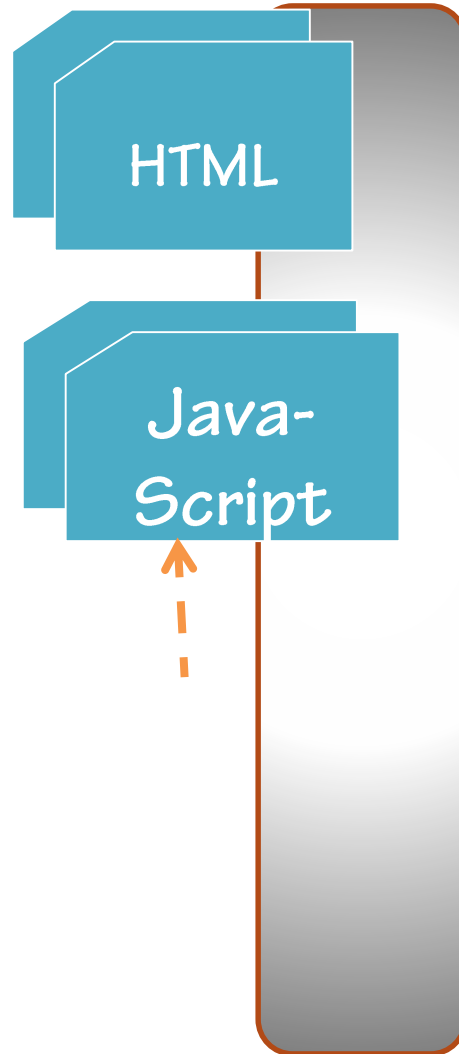
Web Browser

Web Server





Web Browser



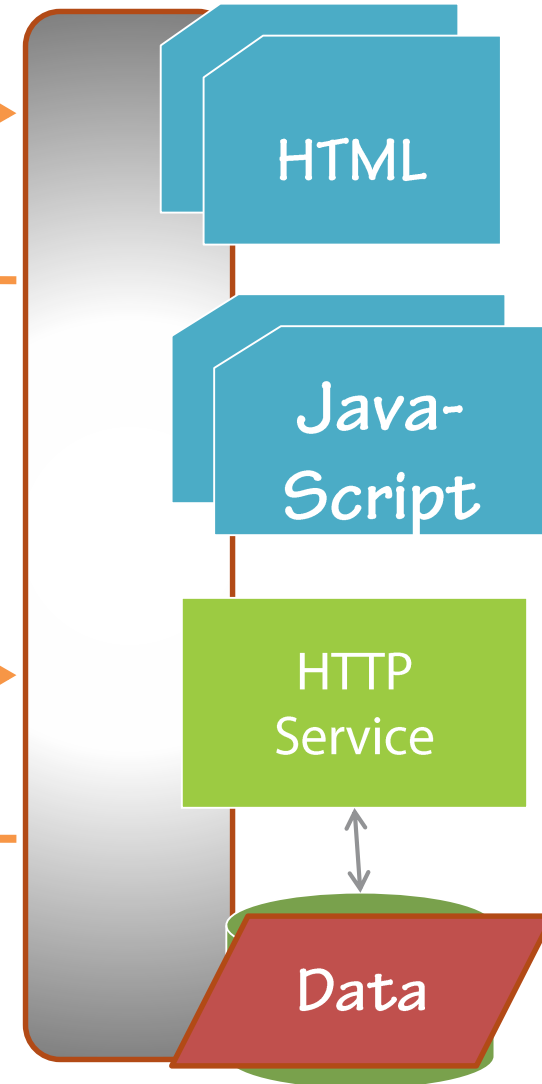
URL Request (www.mysite.com)

Response (Page and Assets)

(http://mysite/api/movies)

Response

Web Server



DOM

- Document Object Model
- Tree structure that defines all of the loaded elements in an HTML document

```
<!DOCTYPE html>
<html>
  <head lang="en">...</head>
  <body ng-app="movieHunter">
    <h1>Movie Hunter</h1>
    <div>...</div>
    <!-- Library Scripts -->
    <!-- Application Script -->
    <!-- Controllers -->
  </body>
</html>
```

DEMO: **ANGULAR APP**

Why Angular?

Expressive
HTML

Powerful
Data Binding

Modularity

Rule-based
Navigation

Easily Call
HTTP Service

Authoritative
Forms

Testable

It's Popular

Movie List View

Expressive
HTML

```
<div ng-controller="MovieListCtrl">
  <div>{{title}}</div>
  <div>Filter by:</div><input type="text" ng-model="listFilter" />

  <div ng-show="listFilter">
    <h3>Movies filtered by: {{listFilter}}</h3>
  </div>
```

```
<table ng-if="movies.length">
  <thead>
    <tr>
      <td><button type="button"
        ng-click="toggleImage()">
        {{showImage ? "Hide" : "Show"}} Poster
      </button></td>
      <td>Title</td> <td>Director</td> <td>Release Date</td> <td>Rating</td>
    </tr>
  </thead>
```

```
<tbody>
  <tr ng-repeat="movie in movies | filter : {title:listFilter} | orderBy : 'title'">
    <td></td>
    <td>{{ movie.title }}</td>
    <td>{{ movie.director }}</td>
    <td>{{ movie.releaseDate | date }}</td>
    <td>{{ movie.mpaa | uppercase }}</td>
  </tr>
</tbody>
</table>
</div>
```

Directives

Expressive
HTML

- Extend HTML
 - Apply special behavior to attributes or elements in the HTML
 - Are simple to define in the HTML
 - Prefixed with "ng-"
- Provide an extensive set of features and capabilities

Directives

```
<body ng-app="movieHunter">  
  <table ng-if="movies.length">  
    <tr ng-repeat="movie in movies | orderBy:'title' ">  
      <img ng-show="showImage"  
          ng-src="" />  
      <button type="button"  
          ng-click="toggleImage()">Poster  
    </button>
```

One Way Data Binding

- View is a projection of the model
 - When the model changes, the view reflects the change
- Binding Expression
 - {{movie.title}}

One Way Data Binding

```
<tr ng-repeat="movie in movies | orderBy:'title' ">
  <td>
    
  </td>
  <td>
    <a>{{movie.title}}</a>
  </td>
  <td>{{movie.director}}</td>
  <td>{{movie.date | date}}</td>
  <td>{{movie.mpaa | uppercase}}</td>
</tr>
```

Two Way Data Binding

- Data in the model and the view are synchronized
 - Changes to the model are reflected in the view
 - Changes in the view are reflected in the model
- Define using a directive: ng-model
 - ng-model="listFilter"

Two Way Data Binding

```
<div>
  <div>Filter by:</div>
  <div>
    <input type="text" ng-model="listFilter" />
  </div>
</div>

<div ng-show="listFilter">
  <h3>Movies filtered by: {{listFilter}}</h3>
</div>

<tr ng-repeat="movie in movies | orderBy:'title'
  | filter : {title : listFilter}">
```

Directives and Binding

- Directives
 - ng-app, ng-if, ng-repeat
 - ng-show/ng-hide
- Data Binding
 - One way binding
 - {{movie.title}}
 - Two way binding
 - ng-model="listFilter"

*Expressive
HTML*

*Powerful
Data Binding*

Angular Code

Modularity

- Write using
 - JavaScript
 - TypeScript
- Divided into
 - Modules
 - Controllers
 - Services

Module

Defines an
Angular
component

Tracks the
component code

Tracks the
dependencies for
the component

Keeps the
application
modularized

Angular Module Method

Setter Method

```
angular.module("movieHunter", ["ngRoute",  
                               "common.services"]);
```

Getter Method

```
angular.module("movieHunter");
```

Angular Main Module

```
var app = angular.module("movieHunter",  
                          []);
```

Angular Main Module

```
function () {  
    var app = angular.module("movieHunter",  
                             []);  
}
```

Angular Main Module - IIFE

```
(function () {  
    var app = angular.module("movieHunter",  
                             []);  
})();
```

The IIFE JavaScript Pattern

- Immediately-Invoked Function Expression
- JavaScript var declarations have global scope
- JavaScript variables and functions defined within a function have local scope

```
(function () {  
    // Code here  
  
})();
```

Module

Modularity

- One "main" module for the application
- Any number of additional modules
 - Group related functionality
 - Define reusable common code

Controller

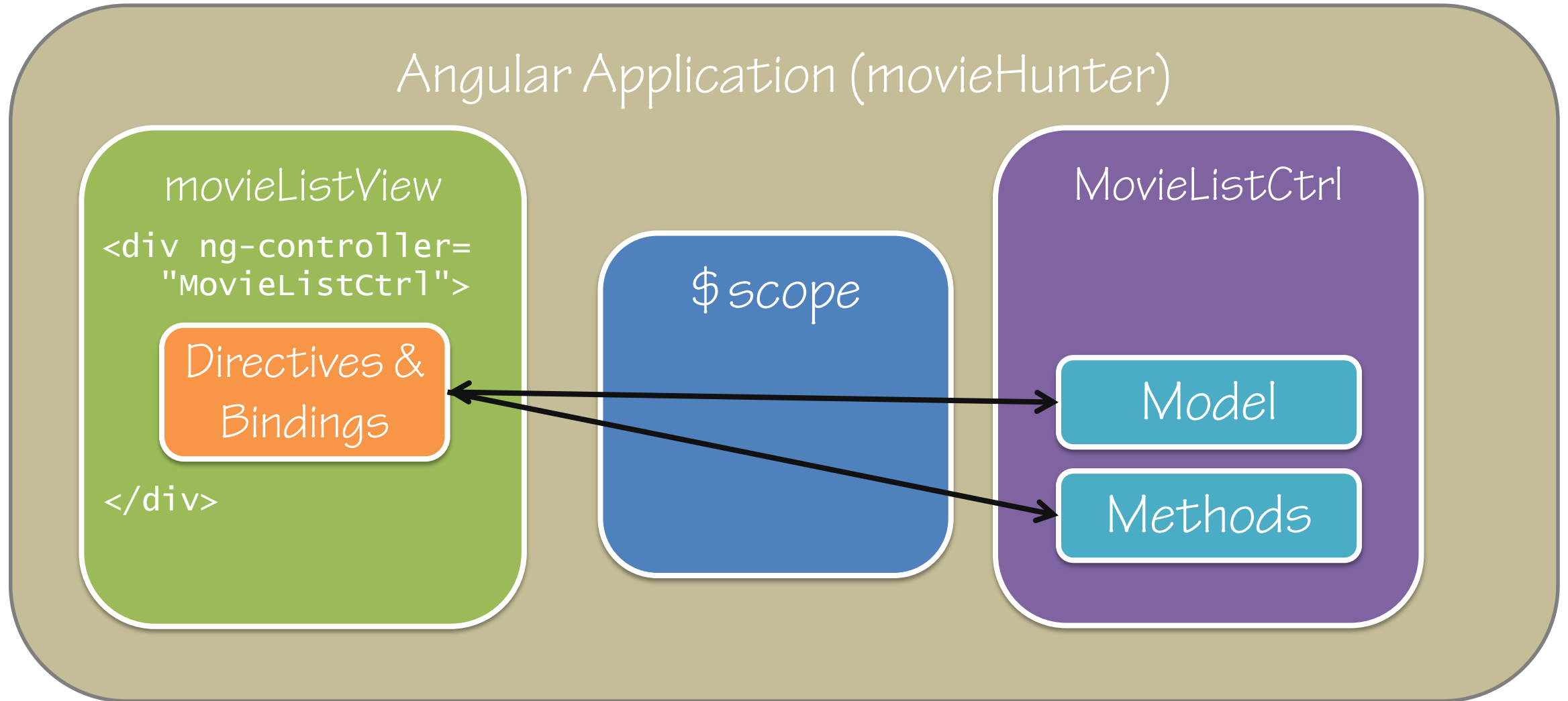
Defines the Model

- Movies

Implements Methods

- Hide/Show Images

View, Controller, and \$scope



Angular Controller

```
(function () {  
  "use strict";  
  
  angular  
    .module("movieHunter")  
    .controller("MovieListCtrl",  
      ["$scope",  
       MovieListCtrl]);  
  
  function MovieListCtrl($scope) {  
    $scope.movies = [];  
    $scope.title = "Search by Movie Title";  
    $scope.showImage = false;  
  
    $scope.toggleImage = function () {  
      $scope.showImage = !$scope.showImage;  
    };  
  }  
})();
```

Angular "Controller As" Syntax

```
(function () {  
    "use strict";  
  
    angular  
        .module("movieHunter")  
        .controller("MovieListCtrl",  
                    ["$scope",  
                     MovieListCtrl]);  
  
    function MovieListCtrl($scope) {  
  
        $scope.movies = [];  
        $scope.title = "Search by Movie Title";  
        $scope.showImage = false;  
  
        $scope.toggleImage = function () {  
            $scope.showImage = !$scope.showImage;  
        };  
  
    }()  
})();
```

Angular "Controller As" Syntax

```
(function () {  
    "use strict";  
  
    angular  
        .module("movieHunter")  
        .controller("MovieListCtrl",  
                    [MovieListCtrl]);  
  
    function MovieListCtrl() {  
  
        $scope.movies = [];  
        $scope.title = "Search by Movie Title";  
        $scope.showImage = false;  
  
        $scope.toggleImage = function () {  
            $scope.showImage = !$scope.showImage;  
        };  
  
    }()  
});
```

Angular "Controller As" Syntax

```
(function () {  
    "use strict";  
  
    angular  
        .module("movieHunter")  
        .controller("MovieListCtrl",  
                    [MovieListCtrl]);  
  
    function MovieListCtrl() {  
        var vm = this;  
        $scope.movies = [];  
        $scope.title = "Search by Movie Title";  
        $scope.showImage = false;  
  
        $scope.toggleImage = function () {  
            $scope.showImage = !$scope.showImage;  
        };  
  
    }()  
})();
```

Angular "Controller As" Syntax

```
(function () {  
    "use strict";  
  
    angular  
        .module("movieHunter")  
        .controller("MovieListCtrl",  
                    [MovieListCtrl]);  
  
    function MovieListCtrl() {  
        var vm = this;  
        vm.movies = [];  
        vm.title = "Search by Movie Title";  
        vm.showImage = false;  
  
        vm.toggleImage = function () {  
            vm.showImage = !vm.showImage;  
        };  
  
    }()  
});
```

View "Controller As" Syntax

```
<div ng-controller="MovieListCtrl">
  <div>{{title}}</div>
  <div>Filter by:</div><input type="text" ng-model="listFilter" />

  <div ng-show="listFilter">
    <h3>Movies filtered by: {{listFilter}}</h3>
  </div>

  <table ng-if="movies.length">
    <thead>
      <tr>
        <td><button type="button"
          ng-click="toggleImage()">
          {{showImage ? "Hide" : "Show"}} Poster
        </button></td>
        <td>Title</td> <td>Director</td> <td>Release Date</td> <td>Rating</td>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="movie in movies | filter : {title:listFilter} | orderBy : 'title'">
        <td></td>
        <td>{{ movie.title }}</td>
        <td>{{ movie.director }}</td>
        <td>{{ movie.releaseDate | date }}</td>
        <td>{{ movie.mpaa | uppercase }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

View "Controller As" Syntax

```
<div ng-controller="MovieListCtrl as vm">
  <div>{{title}}</div>
  <div>Filter by:</div><input type="text" ng-model="listFilter" />

  <div ng-show="listFilter">
    <h3>Movies filtered by: {{listFilter}}</h3>
  </div>

  <table ng-if="movies.length">
    <thead>
      <tr>
        <td><button type="button"
          ng-click="toggleImage()">
          {{showImage ? "Hide" : "Show"}} Poster
        </button></td>
        <td>Title</td> <td>Director</td> <td>Release Date</td> <td>Rating</td>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="movie in movies | filter : {title:listFilter} | orderBy : 'title'">
        <td></td>
        <td>{{ movie.title }}</td>
        <td>{{ movie.director }}</td>
        <td>{{ movie.releaseDate | date }}</td>
        <td>{{ movie.mpaa | uppercase }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

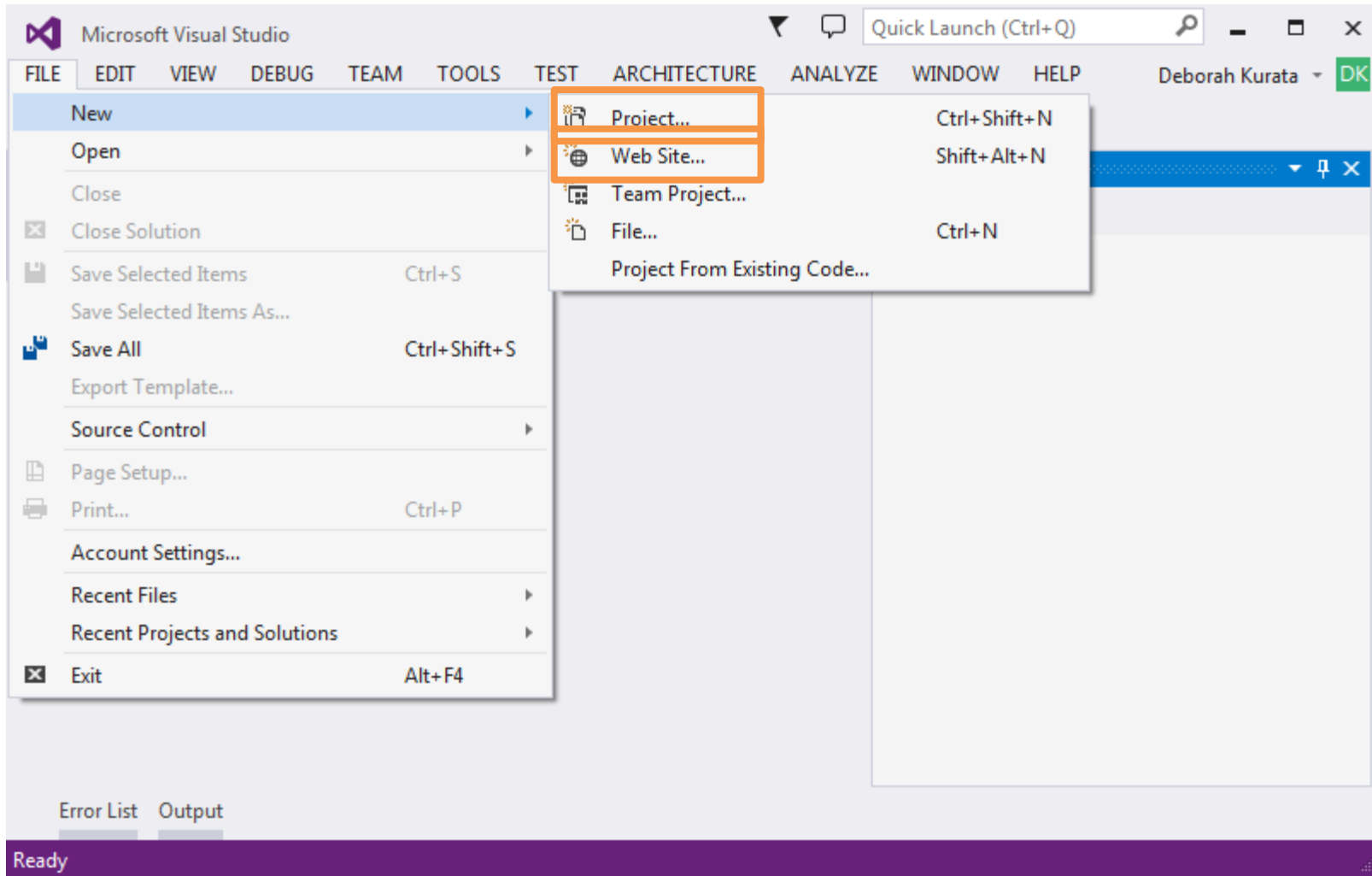
View "Controller As" Syntax

```
<div ng-controller="MovieListCtrl as vm">
  <div>{{vm.title}}</div>
  <div>Filter by:</div><input type="text" ng-model="listFilter" />

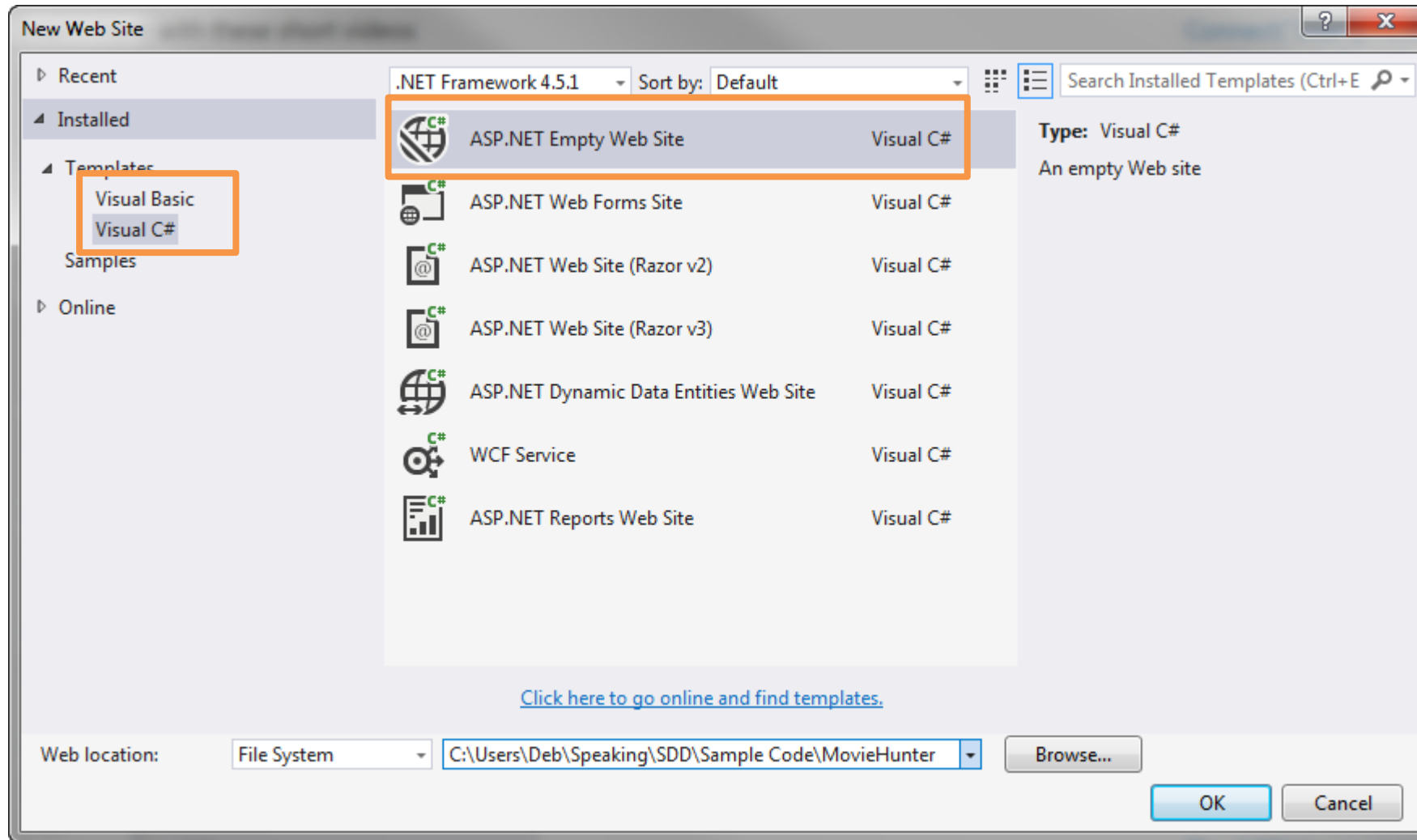
  <div ng-show="listFilter">
    <h3>Movies filtered by: {{listFilter}}</h3>
  </div>

  <table ng-if="vm.movies.length">
    <thead>
      <tr>
        <td><button type="button"
          ng-click="vm.toggleImage()"
          {{vm.showImage ? "Hide" : "Show"}} Poster
        </button></td>
        <td>Title</td> <td>Director</td> <td>Release Date</td> <td>Rating</td>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="movie in vm.movies | filter : {title:listFilter} | orderBy : 'title'">
        <td></td>
        <td>{{ movie.title }}</td>
        <td>{{ movie.director }}</td>
        <td>{{ movie.releaseDate | date }}</td>
        <td>{{ movie.mpaa | uppercase }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

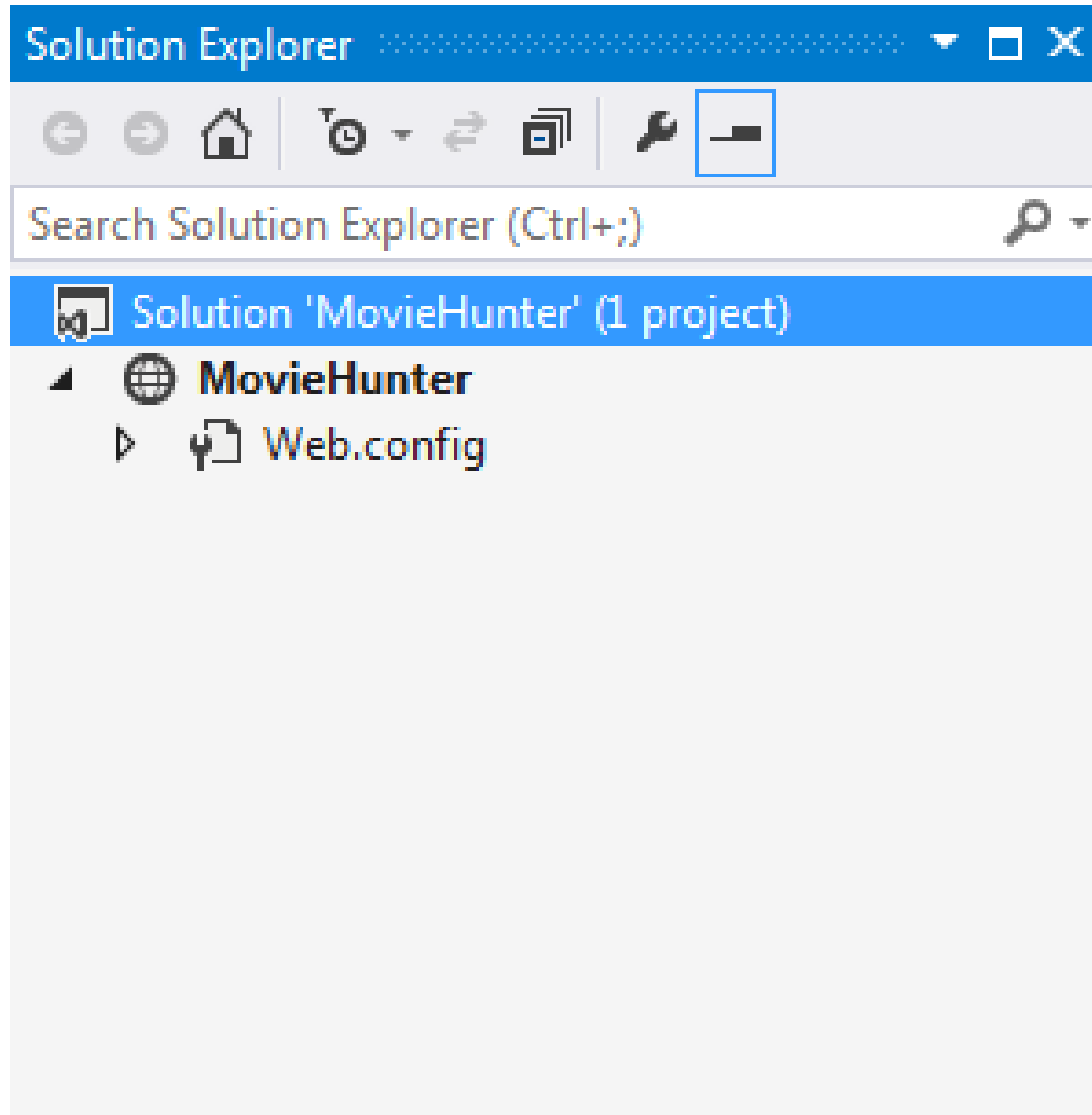

Creating an Angular Project



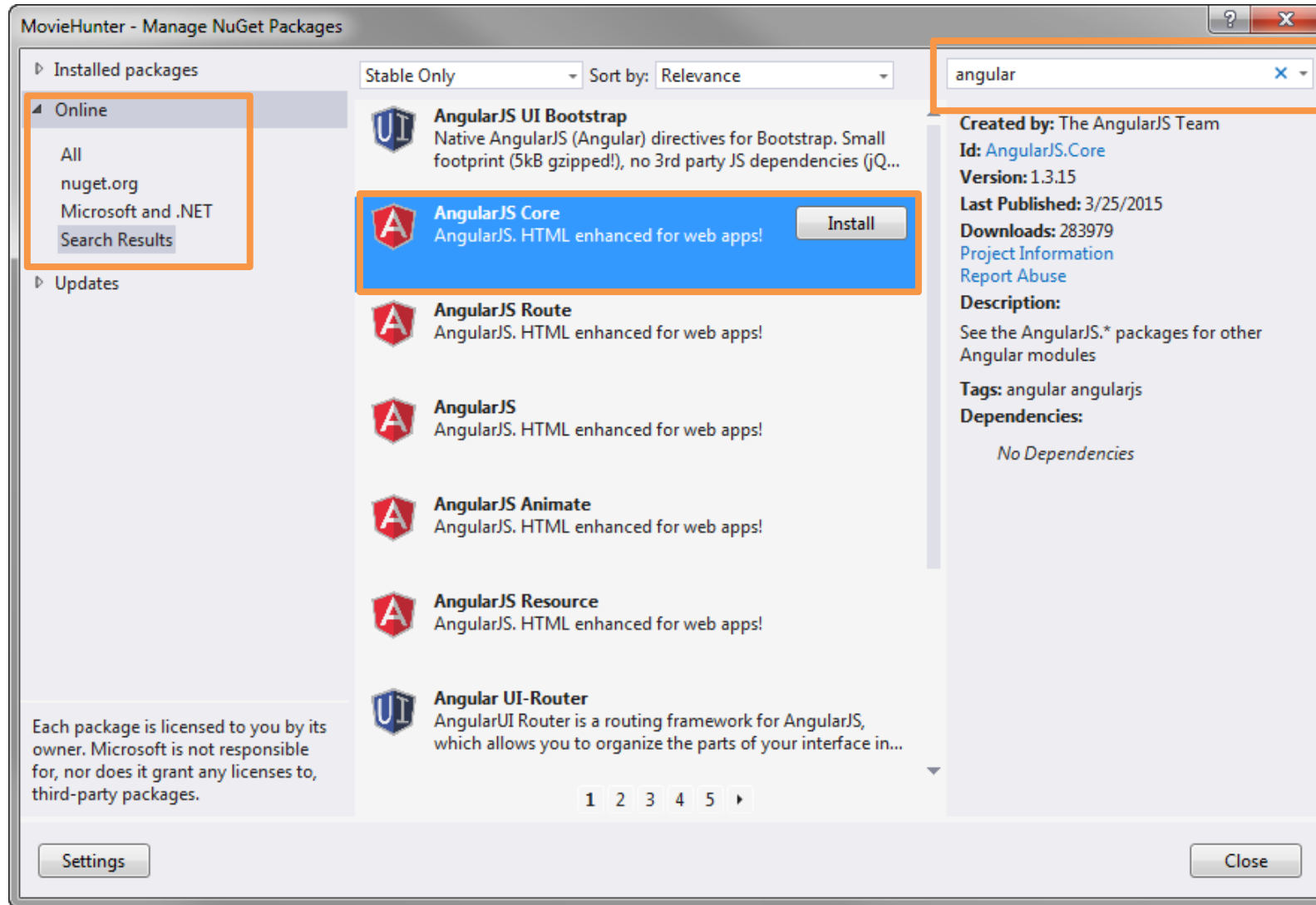
Selecting a Template



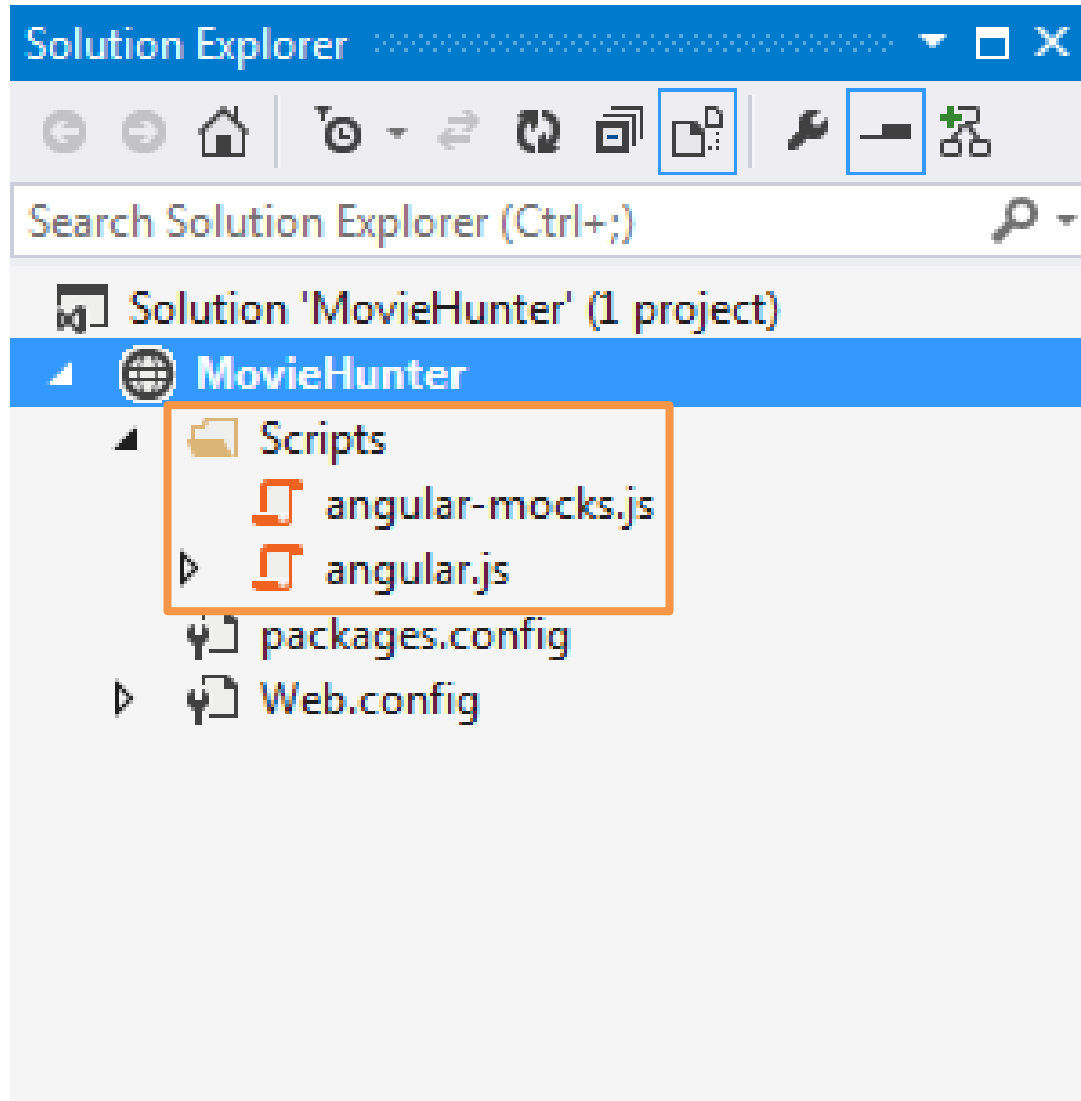
Generated Solution



NuGet Package Manager

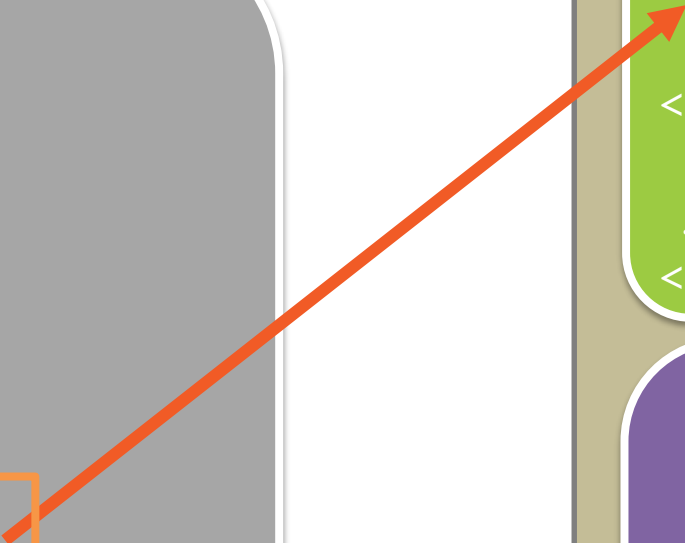


Solution Explorer



index.html

```
<head>
<link .../>
</head>
<body ng-app=
  "movieHunter">
  [menu code here]
  <ng-include=
    "'movieListView.html'">
  <script .../>
</body>
```



app.js (movieHunter)

movieListView.html

```
<div ng-controller=
  "MovieListCtrl as vm">
  ...
</div>
```



movieListCtrl.js (MovieListCtrl)

Model

Methods

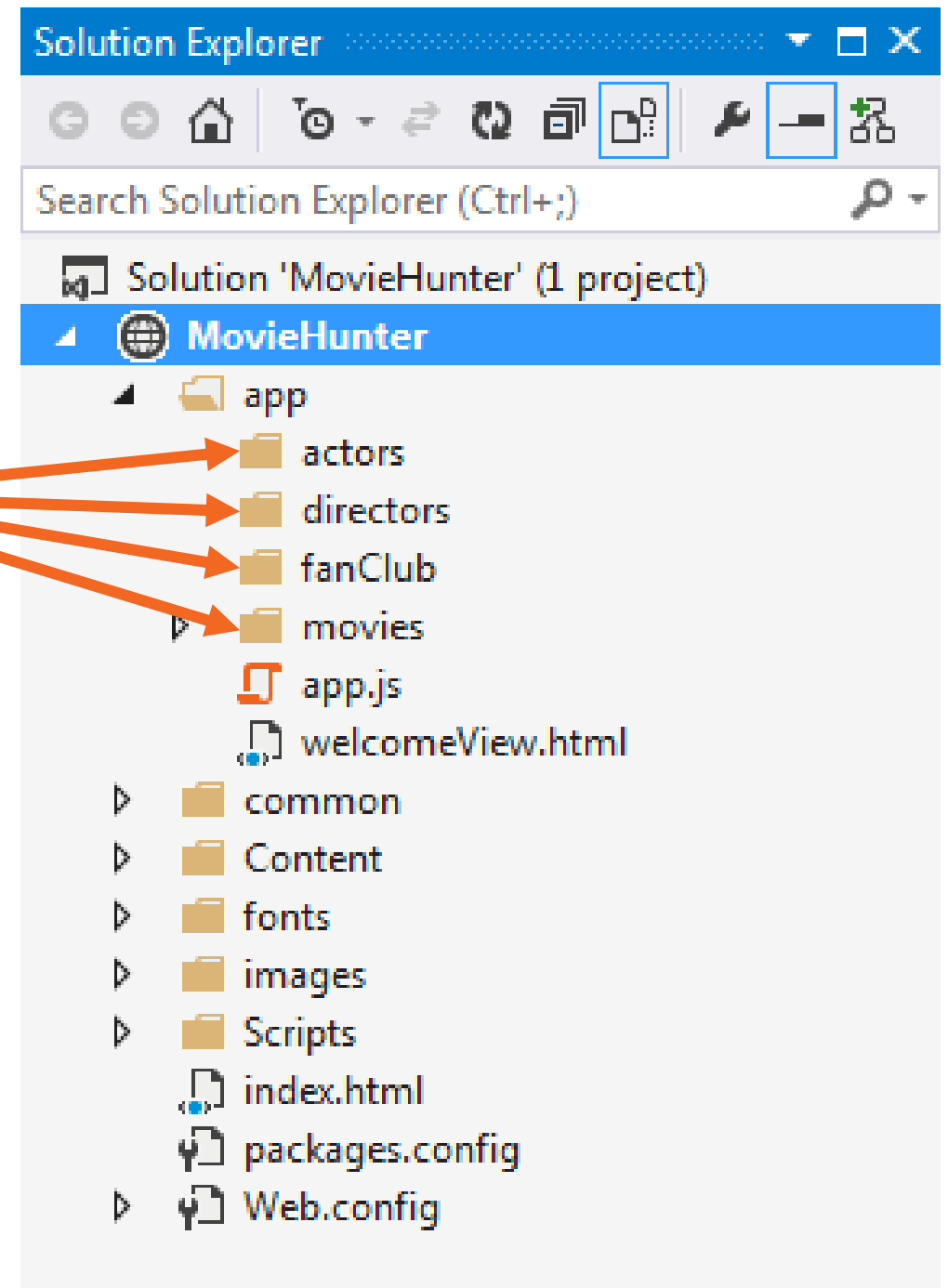
Project Structure

Provides order

Supports multiple
developers

Simplifies Maintenance

By
Feature



Angular Style Guide

<https://github.com/johnpapa/angular-styleguide>

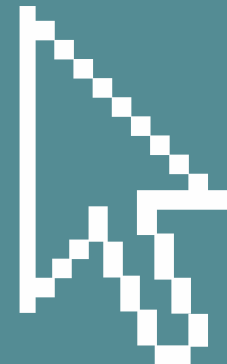
DEMO: HANDS ON LAB 1

Session Materials & Sample Code

<https://github.com/DeborahK/angularu2015>

Hands On Lab 1: Basics

1. Add the script tags
2. Add the director to the HTML table
3. Create the module
4. Create the controller
5. Run the application



Bootstrap

Framework for prettifying
Web pages

Developed by Twitter

Large third party
community

<http://getbootstrap.com>



Bootstrap Grid System

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8								.col-md-4			
.col-md-4				.col-md-4				.col-md-4			
.col-md-6						.col-md-6					

Bootstrap Style Classes

```
<div class="panel panel-primary">
  <div class="panel-heading">
    <h2 class="title">{{vm.title}}</h2>
  </div>

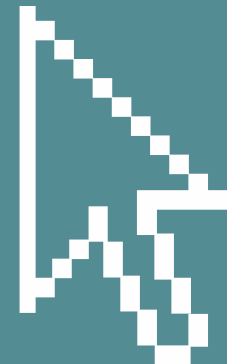
  <div class="panel-body">
    <!-- Filter the Title -->
    <div class="row">
      <div class="col-md-2">Filter by:</div>
      <div class="col-md-4">
        <input type="text" ng-model="listFilter" />
      </div>
    </div>
  </div>
```

DEMO: HANDS ON LAB 2

Hands On Lab 2: Bootstrap

Two options: **Examine** vs **Type**

1. Add the link tag
2. Examine the class attributes
3. Run the application



Review - Basics



Introduction to Angular

Angular and HTML: Building a View

Data Binding

Angular Code: Building the Module & Controller

Bootstrap: Styling the View

Routing

Rate Yourself on **Routing**

- New to Angular - **no** proficiency
- Just starting out - **limited** proficiency
- Doing it but not fully understanding it - **working** proficiency
- Been there, done that, can help others - **full** proficiency

Overview - Routing



Introduction to SPA

Rule-Based Navigation / Routing

Routing Frameworks

Defining Routes (ngRoute)

Navigating Routes (ngRoute)

DEMO: **ANGULAR IN ACTION**

index.html

```
<head>
  <link .../>
</head>
<body ng-app=
  "movieHunter">

  [menu code here]
  <ng-include=
    "'movieListView.html'">

  <script .../>
</body>
```



app.js (movieHunter)

movieListView.html

```
<div ng-controller=
  "MovieListCtrl as vm">
  ...
</div>
```



movieListCtrl.js (MovieListCtrl)

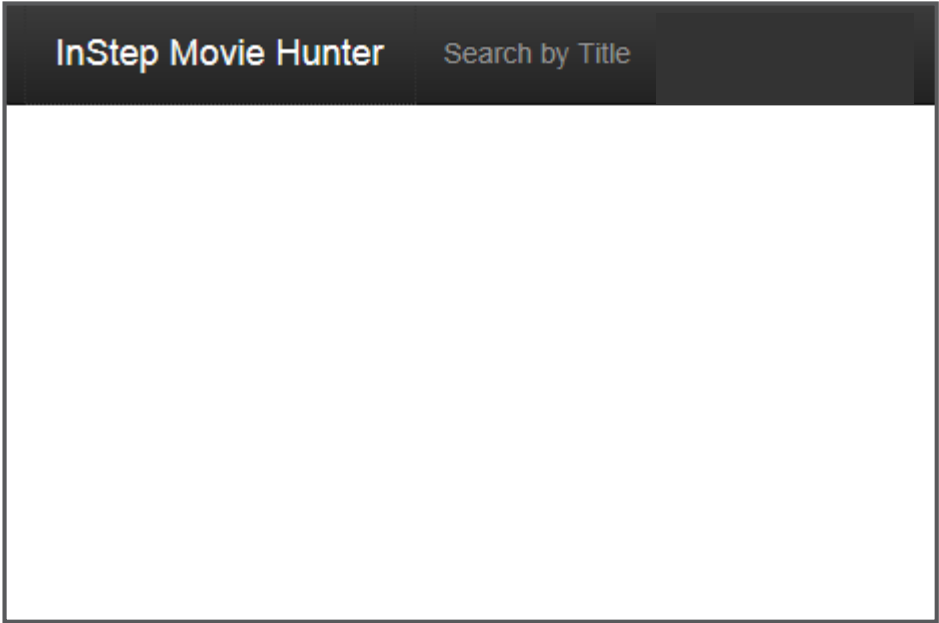
Model

Methods

SPA

- Single Page Application
- Only one page?
 - Only one full HTML page
 - All other pages are HTML fragments called **templates**
 - And are displayed within the one HTML page
- Goals:
 - Enhance user experience
 - Reduce round tripping

Multiple Views



Search by Movie Title				
Filter by:	<input type="text"/>			
Show Poster	Title	Director	Release Date	Rating
	Pirates Of The Caribbean: Curse Of The Black Pearl	Gore Verbinski	Jul 9, 2003	PG-13
	Pirates Of The Caribbean: Dead Man's Chest	Gore Verbinski	Jul 7, 2006	PG-13
	The Lord of the Rings: The Fellowship of the Ring	Peter Jackson	Dec 19, 2001	PG-13
	The Lord of the Rings: The Return of the King	Peter Jackson	Dec 17, 2003	PG-13
	The Lord of the Rings: The Two Towers	Peter Jackson	Dec 18, 2002	PG-13
	The Point	Fred Wolf	Feb 2, 1971	NR

The Fellowship of the Ring

Description: A meek hobbit of the Shire and eight companions set out on a journey to Mount Doom to destroy the One Ring and the dark lord Sauron.

Director: Jackson

Release Date: Dec 19, 2001

Rating: PG-13

Routing

Rule-based
Navigation

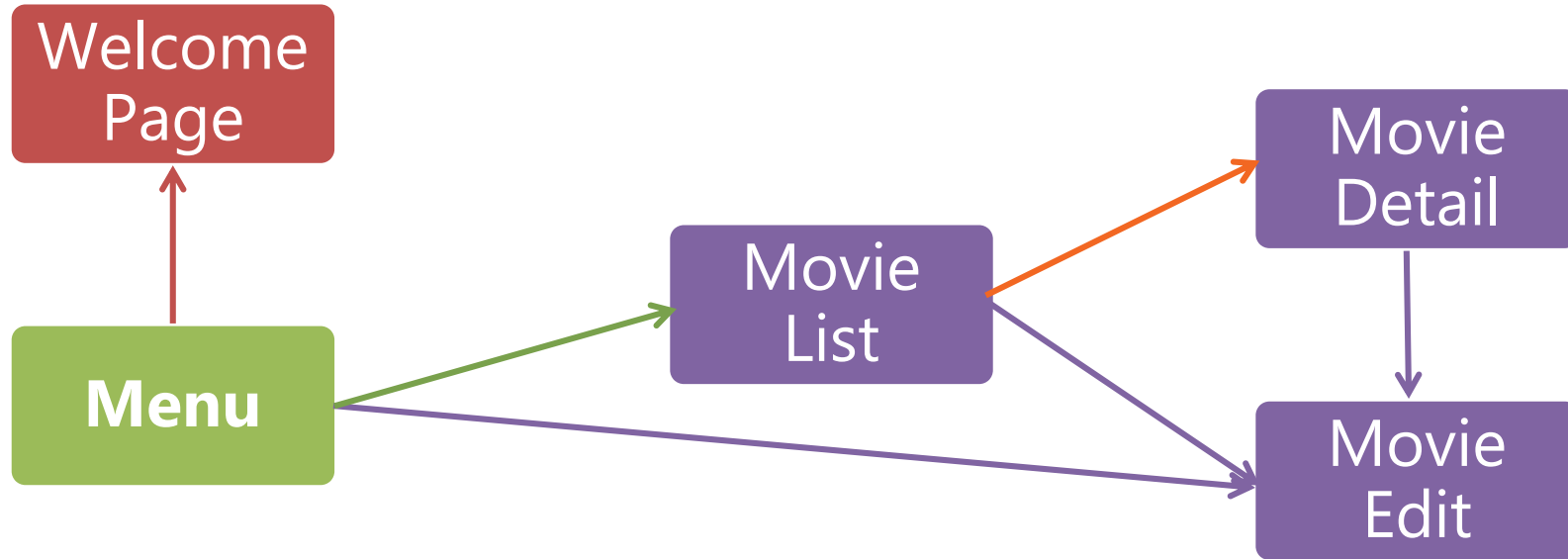
Technique for
navigating
between views

Routes defined
by rules

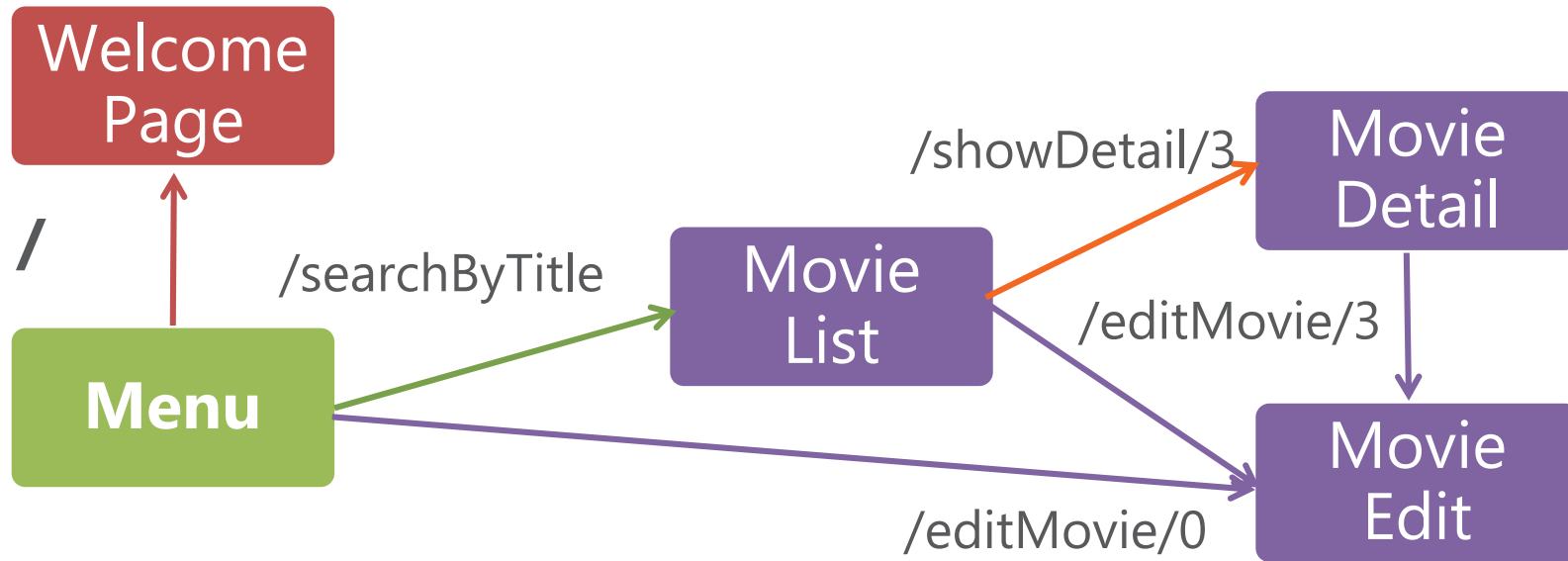
Each route
represents a
specific view

Activating a
route navigates
to that view

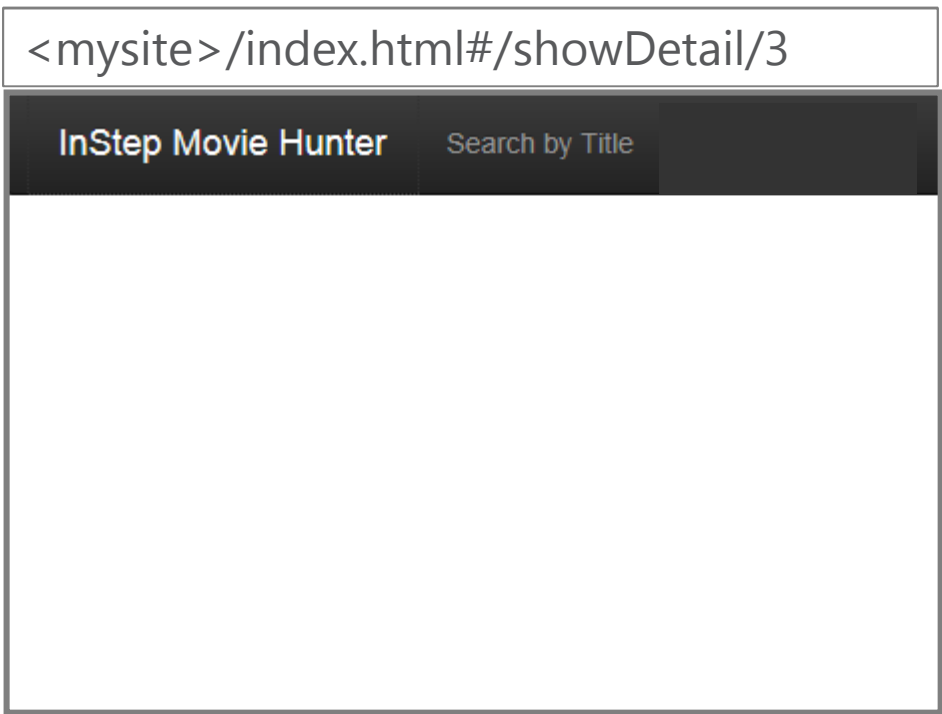
Site Map



Routes



Routing URL Fragments



Search by Movie Title				
Filter by:	<input type="text"/>			
Show Poster	Title	Director	Release Date	Rating
	Pirates Of The Caribbean: Curse Of The Black Pearl	Gore Verbinski	Jul 9, 2003	PG-13
	Pirates Of The Caribbean: Dead Man's Chest	Gore Verbinski	Jul 7, 2006	PG-13
	The Lord of the Rings: The Fellowship of the Ring	Peter Jackson	Dec 19, 2001	PG-13
	The Lord of the Rings: The Return of the King	Peter Jackson	Dec 17, 2003	PG-13
	The Lord of the Rings: The Two Towers	Peter Jackson	Dec 18, 2002	PG-13
	The Point	Fred Wolf	Feb 2, 1971	NR

The Fellowship of the Ring

Description: A meek hobbit of the Shire and eight companions set out on a journey to Mount Doom to destroy the One Ring and the dark lord Sauron.

Director: Jackson

Release Date: Dec 19, 2001

Rating: PG-13

Routing Frameworks

ngRoute

UI Router



Setting Up Routing: ngRoute

Define where each view should appear



Set up the routing rules:

- Path, associated view, associated controller

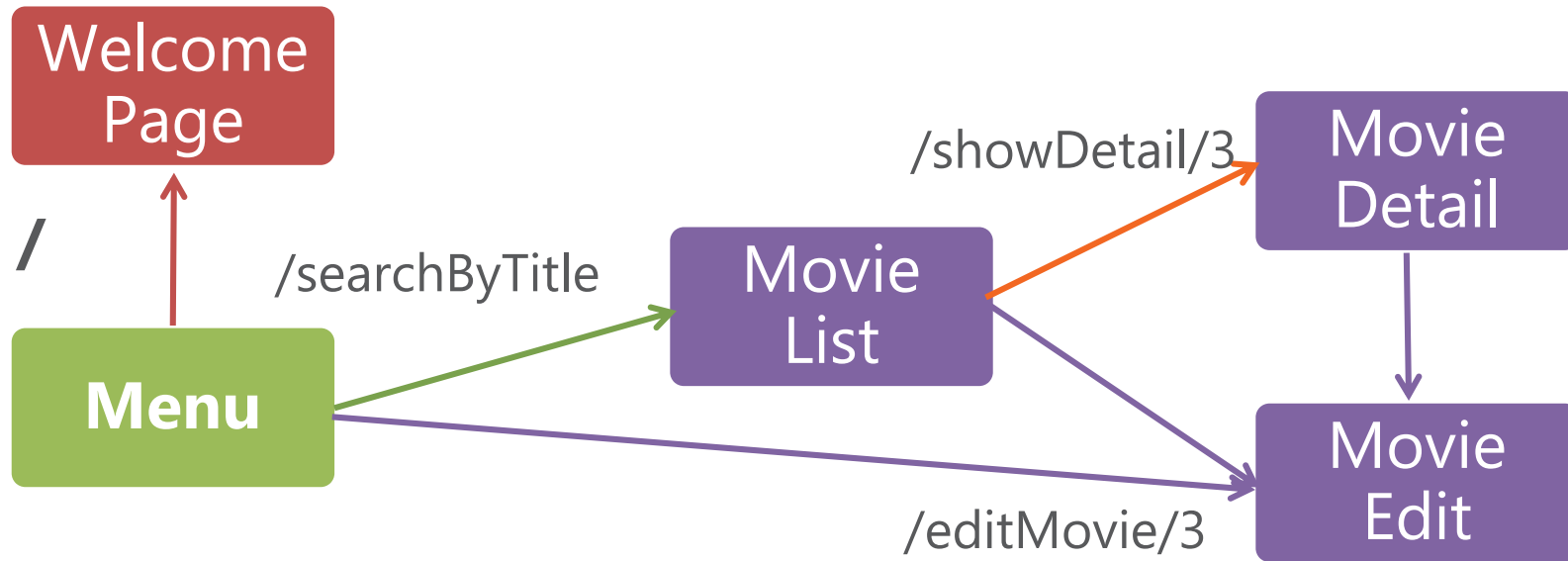


Navigate by activating a route

ngRoute: index.html

```
<div class="container">  
  <div ng-view></div>  
</div>
```

Routes



ngRoute: app.js

```
var app = angular.module("movieHunter", ["ngRoute"]);

app.config(['$routeProvider',
  function ($routeProvider) {
    $routeProvider
      .when("/", {
        templateUrl: "app/welcomeview.html"
      })
      .when("/searchByTitle", {
        templateUrl: "app/movies/movieListView.html",
        controller: "MovieListCtrl as vm"
      })
      .when("/showDetail/:movieId", {
        templateUrl: "app/movies/movieDetailView.html",
        controller: "MovieDetailCtrl as vm"
      })
      .otherwise("/");
  }
]);
```

Navigating Routes

- Anchor tags
 - Link to the appropriate route

```
<a ng-href="#searchByTitle">
```
- \$location service

```
$location.path("/searchByTitle");
```

ngRoute: index.html

```
<div class="navbar-collapse collapse">  
  <ul class="nav navbar-nav">  
    <li>  
      <a ng-href="#searchByTitle">Search by Title</a>  
    </li>  
  </ul>  
</div>
```

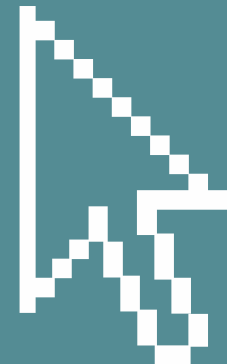
ngRoute: movieDetailView.html

```
<div class="panel-footer">  
  <a class="btn btn-default"  
    ng-href="#searchByTitle"  
    style="width:80px">  
    <i class="glyphicon glyphicon-chevron-left"></i>  
    Back  
  </a>  
</div>
```

DEMO: HANDS ON LAB 3

Hands On Lab 3: Routing

1. Add the script tags
2. Replace ng-include with ng-view
3. Add ngRoute as a dependency
4. Examine the routing code!
5. Add the ng-href to activate the #searchByTitle route to return from the movieDetailView page
6. Run the application



Review - Routing



Introduction to SPA

Rule-Based Navigation / Routing

Routing Frameworks

Defining Routes (ngRoute)

Navigating Routes (ngRoute)

Data Access

Rate Yourself on **Data Access**

- New to Angular - **no** proficiency
- Just starting out - **limited** proficiency
- Doing it, not fully understanding it - **working** proficiency
- Been there, done that, can help others - **full** proficiency

Overview – Data Access



Request/Response Flow

JavaScript Promises

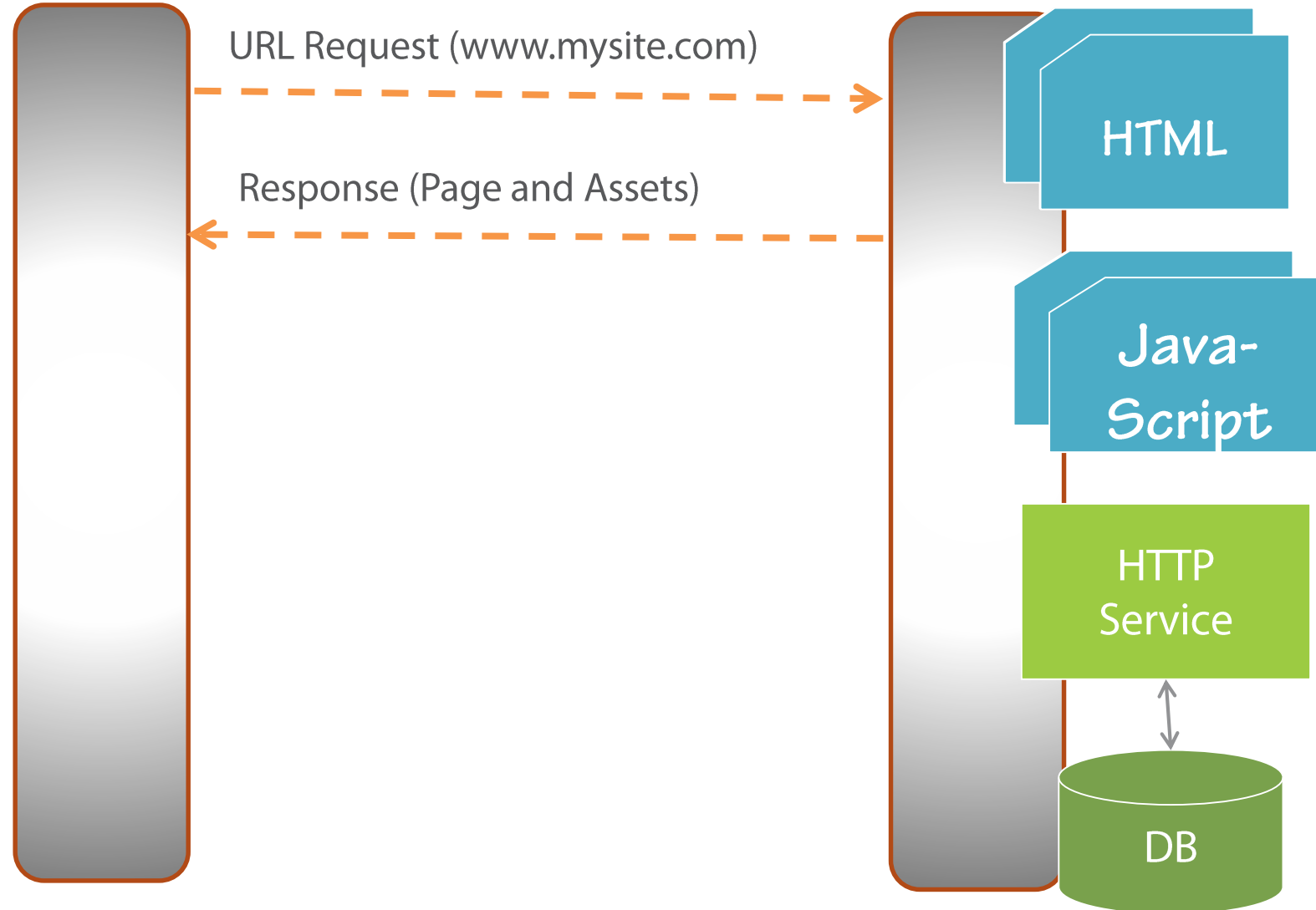
Retrieving Data via HTTP Service

Reusable Code: Building a Service



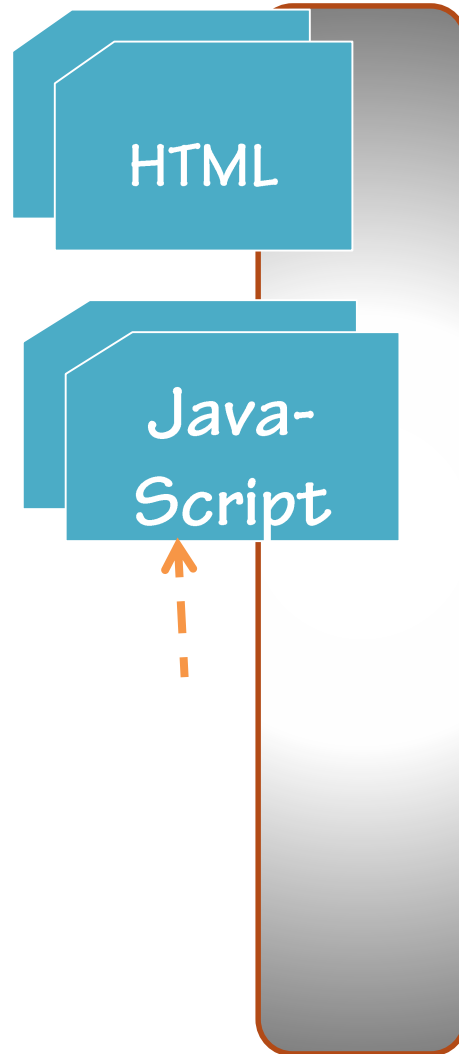
Web Browser

Web Server





Web Browser



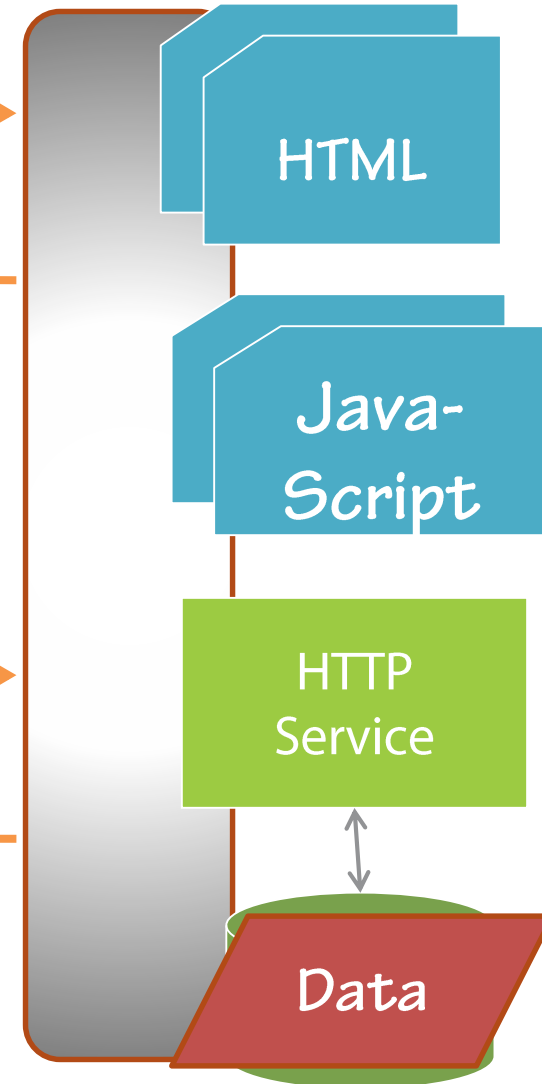
URL Request (www.mysite.com)

Response (Page and Assets)

(http://mysite/api/movies)

Response

Web Server



Retrieving Data

InStep Movie Hunter

Search by Title

Search by Movie Title

Filter by:

Show Poster

Title

Director

Release Date

Rating

GET Request
api/movies

HTTP
Service

Retrieving Data

The screenshot shows a web application titled "InStep Movie Hunter". It has a dark header with the title and a "Search by Title" button. Below the header, there is a search bar with the placeholder text "Search by Movie Title". To the left of the search bar is a "Filter by:" label and a text input field. Below the search bar, there is a "Show Poster" button. The main content area displays a table with columns for "Title", "Director", "Release Date", and "Rating". The table is currently empty.

GET Request

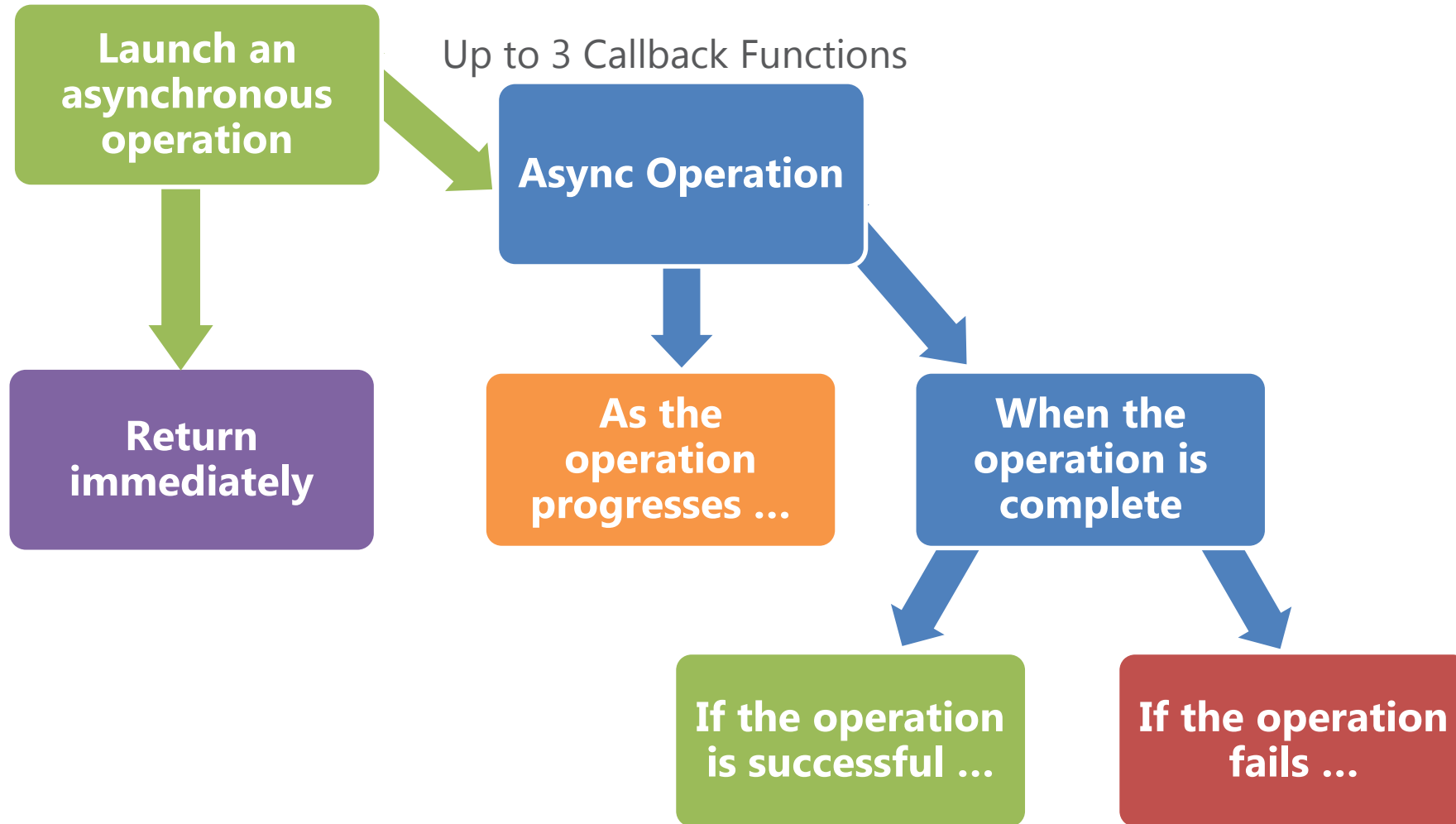
api/movies

Response

HTTP
Service

```
[
  { director: "Peter Jackson",
    movieId: 1,
    mpaa: "pg-13",
    releaseDate: "2001-12-19T00:00:00",
    title: "The Lord of the Rings: The Fellowship of the Ring"
  },
  { director: "Peter Jackson",
    movieId: 2,
    mpaa: "pg-13",
    releaseDate: "2002-12-18T00:00:00",
    title: "The Lord of the Rings: The Two Towers"
  },
  ...
]
```

JavaScript Promise



Calling HTTP Service

\$http

\$resource

\$httpBackend


```
$http.get("/api/movies/")  
  .success(function(response) {  
    vm.movies = response.data;  
  });
```

\$http

Built into Angular Core

Facilitates communication with a back-end service
get, post, put, delete, ...

Asynchronous

REST

- REpresentational S tate T ransfer
- Requests and responses involve the transfer of resources
- Resources are identified with a URL
 - /api/movies/
- Requests utilize a standard set of HTTP verbs
 - GET | POST | PUT | DELETE
- And much more!

```
function movieResource($resource) {  
    return $resource("/api/movies/:movieId")  
}
```

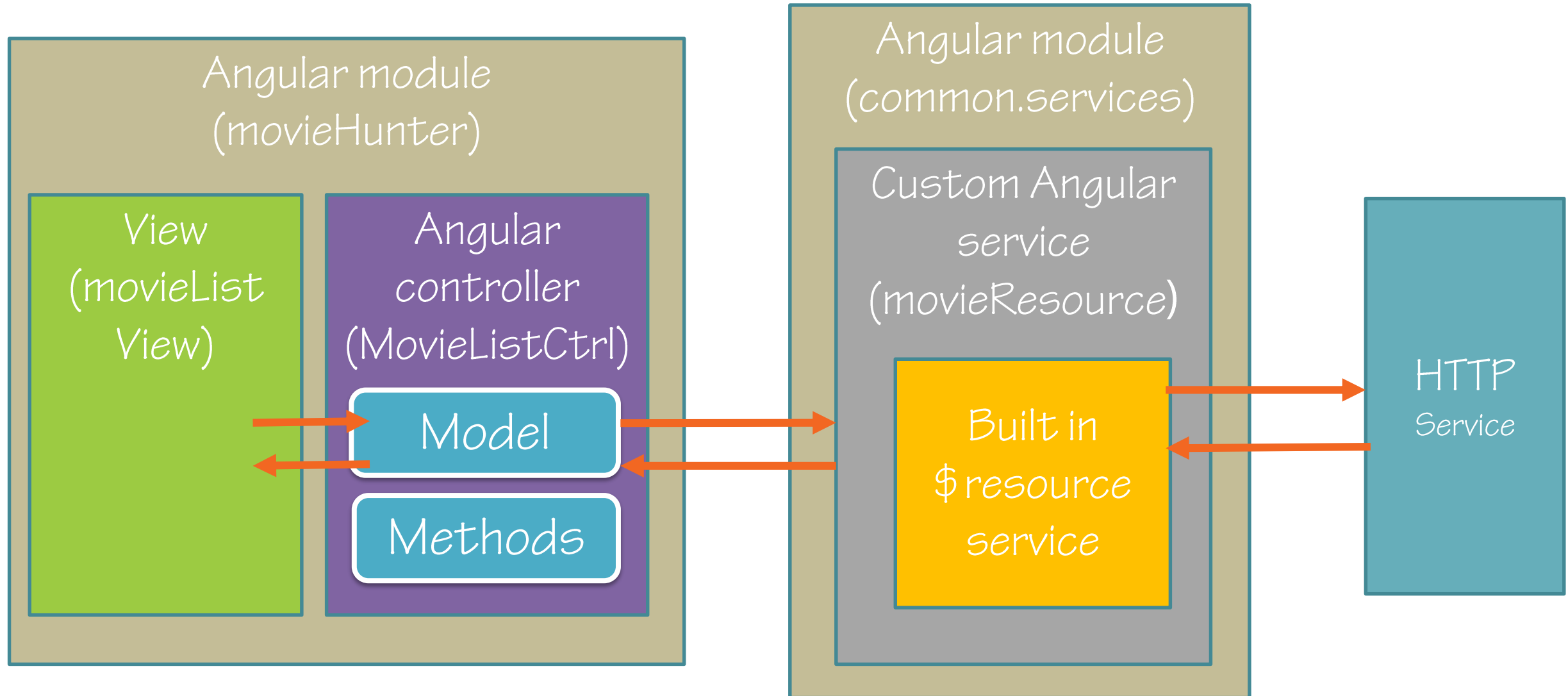
\$resource

Separate Angular component: angular-resource
ngResource

Abstraction on top of \$http for calling RESTful services

Requires less code

Reusable Data Access Service



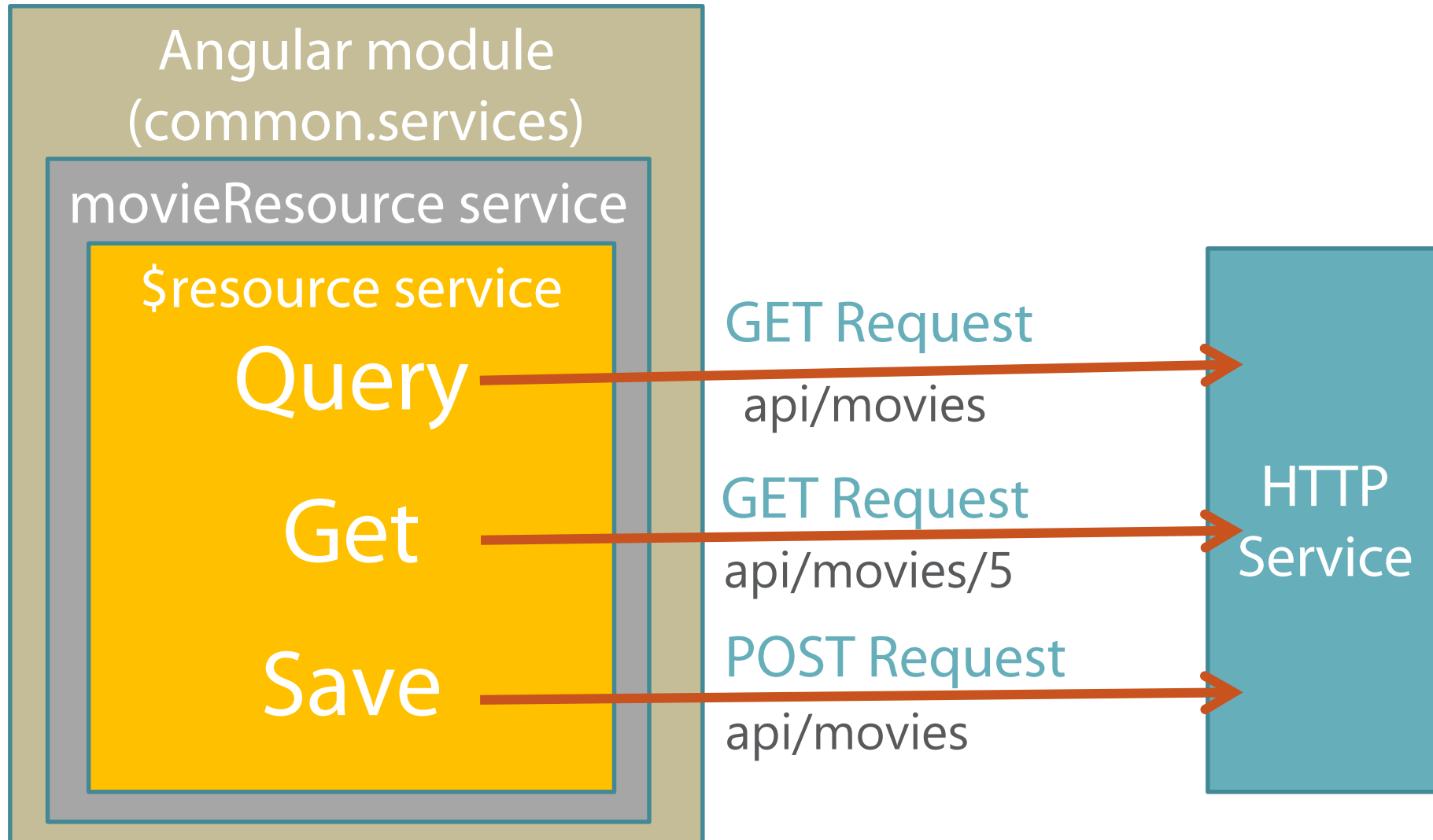
Module: common.services

```
(function () {  
    "use strict";  
  
    angular  
        .module("common.services",  
                ["ngResource"])  
        .constant("appSettings",  
            {  
                serverPath : "http://localhost:1561"  
            });  
})();
```

Service: movieResource

```
(function () {  
    "use strict";  
  
    angular  
        .module("common.services")  
        .factory("movieResource",  
            ["$resource",  
             "appSettings",  
             movieResource]);  
  
    function movieResource($resource, appSettings) {  
        return $resource(appSettings.serverPath +  
                           "/api/movies/:id");  
    }  
}());
```

Built-In \$resource Methods



\$resource Methods

Method Name	Verb	URL	Description
query	GET	/api/movies	Retrieves all of the movies
get	GET	/api/movies/5	Retrieves the specified movie
save	POST	/api/movies	Saves modifications to the movie

```
movieResource.query(function (data) {  
    vm.movies = data;  
});
```

```
movieResource.get({ movieId: $routeParams.movieId },  
    function (data) {  
        vm.movie = data;  
    });
```


\$resource Methods (cont)

Method Name	Verb	URL	Description
query	GET	/api/movies	Retrieves all of the movies
get	GET	/api/movies/5	Retrieves the specified movie
save	POST	/api/movies	Saves modifications to the movie

```
vm.movie.$save(function (data) {  
    vm.message = "Save successful.";  
});
```

Using movieResource

```
(function () {  
    "use strict";  
  
    angular  
        .module("movieHunter")  
        .controller("MovieListCtrl",  
                    ["movieResource",  
                     MovieListCtrl]);  
  
    function MovieListCtrl(movieResource) {  
        var vm = this;  
        vm.movies = [];  
        vm.title = "Search by Movie Title";  
        vm.showImage = false;  
    }  
}
```

Using movieResource (cont)

```
vm.toggleImage = function () {  
    vm.showImage = !vm.showImage;  
};
```

```
movieResource.query(  
    function (data) {  
        vm.movies = data;  
    });
```

```
}
```

```
})();
```

Data Access

InStep Movie Hunter

Search by Title

Search by Movie Title

Filter by:

Show Poster

Title

Director

Release Date

Rating

GET Request
api/movies

HTTP
Service

Data Access

InStep Movie Hunter

Search by Title

Search by Movie Title

Filter by:

Show Poster

Title

Director

Release Date

Rating

GET Request

api/movies

Response

HTTP
Service

```
[  
  { director: "Peter Jackson",  
    movieId: 1,  
    mpaa: "pg-13",  
    releaseDate: "2001-12-19T00:00:00",  
    title: "The Lord of the Rings: The Fellowship of the Ring"  
  },  
  { director: "Peter Jackson",  
    movieId: 2,  
    mpaa: "pg-13",  
    releaseDate: "2002-12-18T00:00:00",  
    title: "The Lord of the Rings: The Two Towers"  
  },  
  ...  
]
```

Mocking Data Access

api/movies

InStep Movie Hunter Search by Title

Search by Movie Title

Filter by:

Show Poster

Title	Director	Release Date	Rating
-			
-			
-			
-			
-			
-			



Mocking
Service

HTTP
Service

Mocking Data Access

api/movies

InStep Movie Hunter Search by Title

Search by Movie Title

Filter by:

Show Poster

Title	Director	Release Date	Rating
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-

Mocking
Service

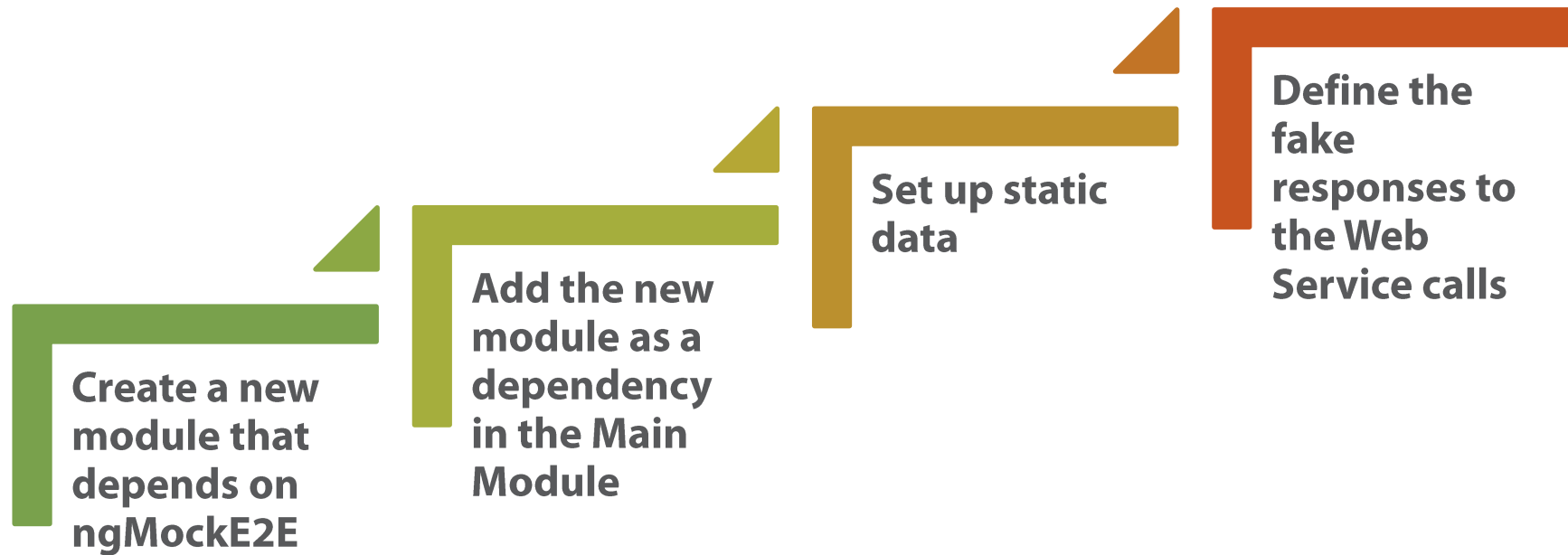
```
[
  { director: "Peter
Jackson",
    movieId: 1,
    mpaa: "pg-13",
    releaseDate: "2001-12-
19T00:00:00",
    title: "The Lord of the
Rings: The Fellowship of the
Ring"
  },
  ...
]
```

HTTP
Service

\$httpBackend

- Angular's fake HTTP backend implementation
- Mocks the calls to the Web Service
- Returns static data to the application
- Two implementations:
 - **ngMock**: for unit testing applications
 - **ngMockE2E**: for end-to-end testing or backend-less development

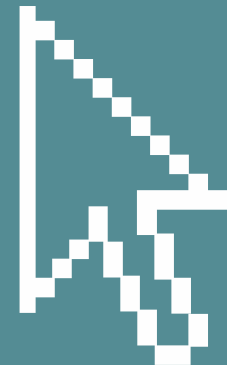
Steps to Mocking the Web Server



DEMO: HANDS ON LAB 4

Hands On Lab 4: Data Access

1. Add the script tags
2. Add a dependency on the new modules
3. Examine the new movieResource service
4. Inject the service into the movieListCtrl
5. Call the \$resource query method
6. Inject the service into the movieDetailCtrl
7. Call the \$resource get method
8. Run the application



Review – Data Access



Request/Response Flow

JavaScript Promises

Retrieving Data via HTTP Service

Reusable Code: Building a Service

Forms & Validation

Rate Yourself on **Forms and Validation**

- New to Angular - **no** proficiency
- Just starting out - **limited** proficiency
- Doing it but not fully understanding it - **working** proficiency
- Been there, done that, can help others - **full** proficiency

Overview – Forms & Validation



Building a Data Entry Form

Performing Validation

Setting Validation Styles

Displaying Validation Messages

DEMO: FORMS AND VALIDATION


```
<form name="movieForm">  
  <fieldset>  
    <legend>Edit Movie Information</legend>  
  </fieldset>  
</form>
```

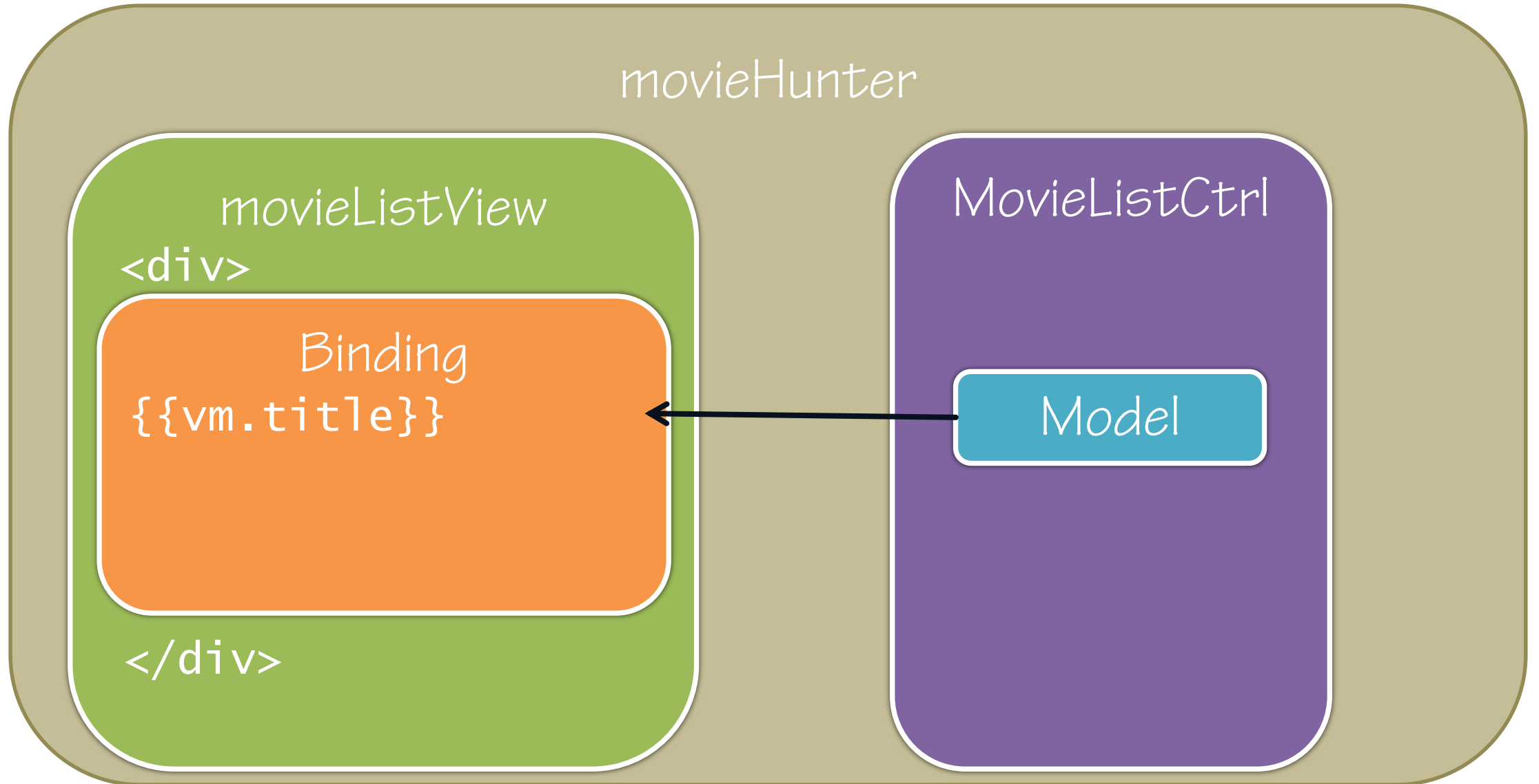
Creating a Form

It's just HTML

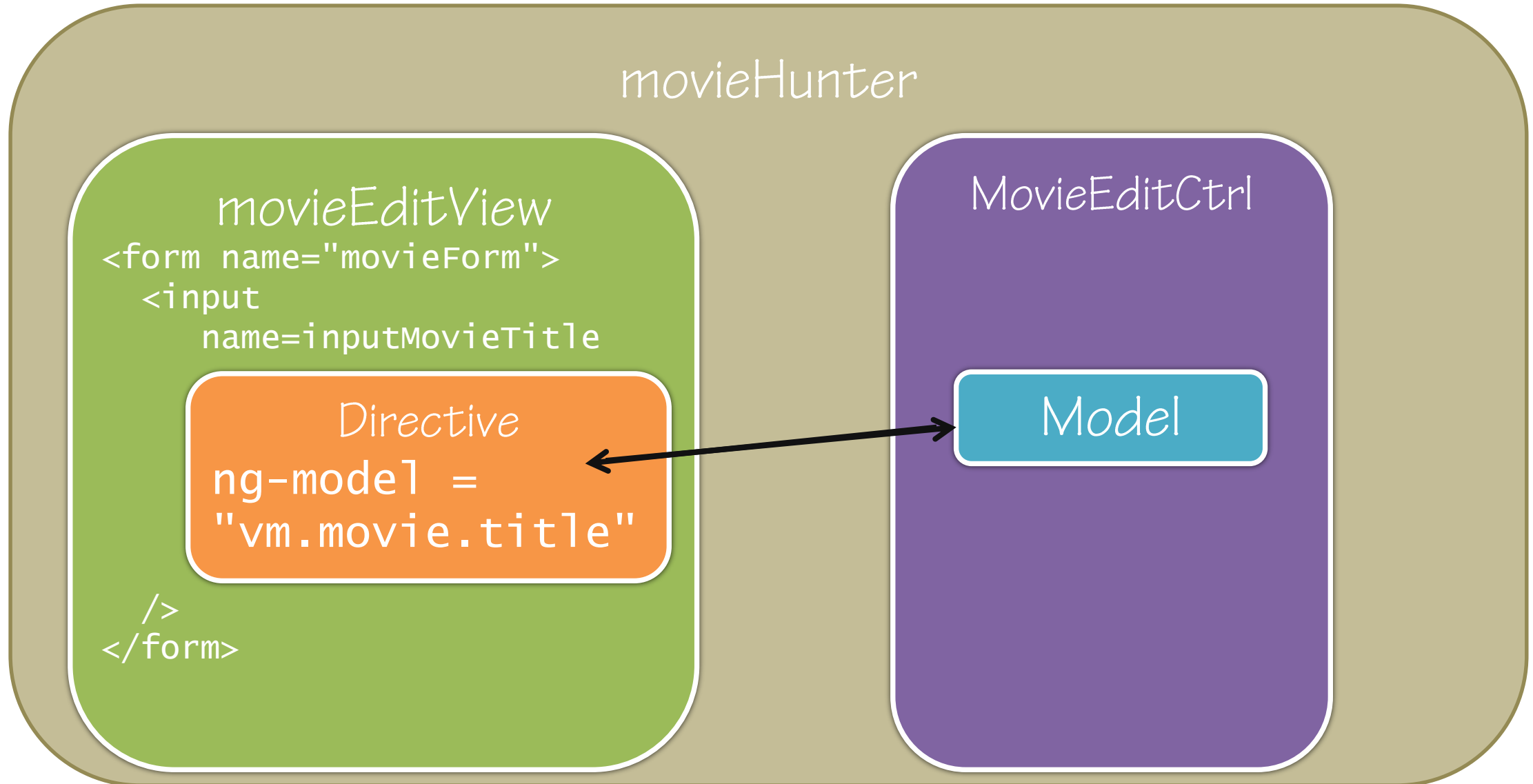
Data Binding

- One-Way Binding
 - View is the projection of the model
 - When the model changes, the view reflects the change
- Two-Way Binding
 - View and model data are synchronized
 - Changes to the model are reflected in the view
 - Changes in the view are reflected in the model

One-Way Binding



Two-Way Binding



Two-Way Binding

```
<div>
  <label for="inputMovieTitle">
    Movie Title</label>
  <input id="inputMovieTitle"
    type="text"
    placeholder="Movie Title" />
</div>
<div>
  <label for="inputDescription">
    Description</label>
  <textarea id="inputDescription"
    placeholder="Description"
    rows="3" />
</div>
```

Two-Way Binding

```
<div>
  <label for="inputMovieTitle">
    Movie Title</label>
  <input id="inputMovieTitle"
    type="text"
    placeholder="Movie Title"
    ng-model="vm.movie.title" />
</div>
<div>
  <label for="inputDescription">
    Description</label>
  <textarea id="inputDescription"
    placeholder="Description"
    rows="3"
    ng-model="vm.movie.description" />
</div>
```

The Result

InStep Movie Hunter

Search by Title

+ Add Movie

Edit Movie Information

Movie Title

Description

While Frodo and
Sam edge closer
to Mordor with the

Bootstrap Styles

```
<div class="form-group">
  <label class="col-md-2 control-label"
    for="inputMovieTitle">
    Movie Title</label>
  <div class="col-md-6">
    <input class="form-control"
      id="inputMovieTitle"
      type="text"
      placeholder="Movie Title"
      ng-model="vm.movie.title" />
  </div>
</div>
```

The Result

InStep Movie Hunter

Search by Title

+ Add Movie

The Lord of the Rings: The Two Towers

Edit Movie Information

Movie Title

The Lord of the Rings: The Two Towers

Description

While Frodo and Sam edge closer to Mordor with the help of the shifty Gollum, the divided fellowship makes a stand against Sauron's new ally, Saruman, and his hordes of Isengard.

Save

Cancel

DEMO:
HANDS ON LAB 5-1

Hands On Lab 5: Forms

1. Add the script tags
2. Add an editMovie route for the movieEditView
3. Run the application
4. Add a label and input box for the movie director
5. Run the application



Validation

InStep Movie Hunter

Search by Title

+ Add Movie

The Lord of the Rings: The Two Towers

Edit Movie Information

Movie Title

Movie title is required.

Description

While Frodo and Sam edge closer to Mordor with the help of the shifty Gollum, the divided fellowship makes a stand against Sauron's new ally, Saruman, and his hordes of Isengard.

Save

Cancel

Prepare the form

**Ensure the
form has a
name**

**Set the
novalidate
attribute**

```
<form class="form-horizontal"  
      name="movieForm">  
  novalidate>
```

Angular Validation Techniques

Type Attribute

```
type="email"
```

Required Attribute

```
required
```

HTML Validation Attributes

```
min = "1"
```

Angular Validation
Attributes

```
ng-minlength = "3"
```

Custom Angular Directive

```
my-directive=""
```

Validation Attributes

```
<div class="col-md-8">  
  <input class="form-control"  
    id="inputMovieTitle"  
    name="inputMovieTitle"  
    type="text"  
    placeholder="Title (required)"  
    ng-model="vm.movie.title"  
    required  
    ng-minlength="3"  
    ng-maxlength="50" />  
</div>
```


Marking Invalid Elements

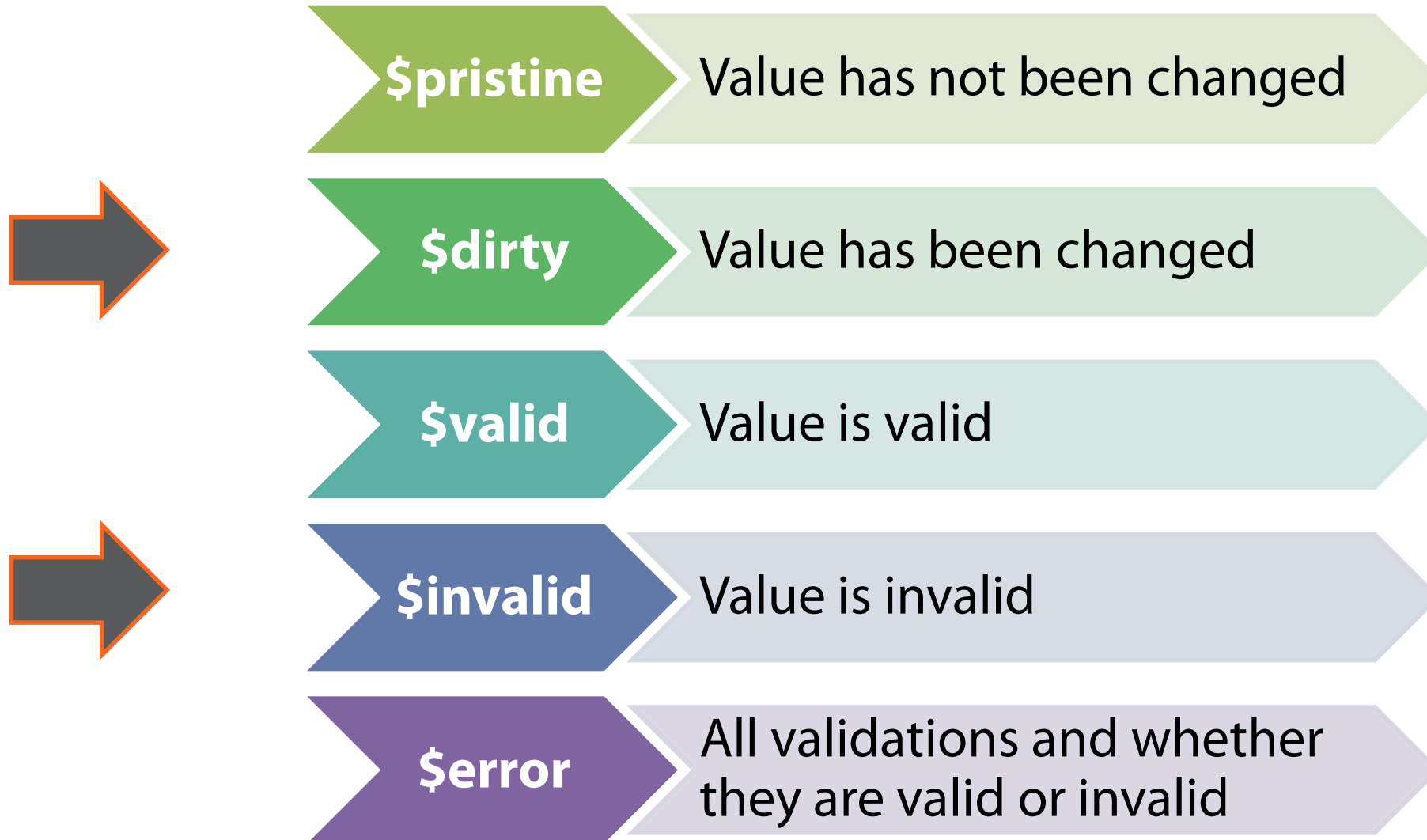
**ngClass
Directive**

**Bootstrap
Style**

**Angular
Validation
States**

```
<div class="form-group"  
  ng-class="{ 'has-error':  
    movieForm.inputMovieTitle.$invalid &&  
    movieForm.inputMovieTitle.$dirty }">  
  
</div>
```

Angular Validation States



Angular Validation States (New in 1.3)

\$touched

Control was "touched" (on blur)

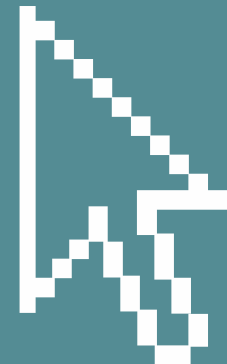
\$untouched

Control never "touched" (blur)

DEMO: HANDS ON LAB 5-2

Hands On Lab 5: Validation

1. Use attributes to define appropriate validation rules
2. Add the ng-class attribute to set the validation color coding
3. Run the application



Validation Messages

InStep Movie Hunter

Search by Title

+ Add Movie

The Lord of the Rings: The Two Towers

Edit Movie Information

Movie Title

Description

While Frodo and Sam edge closer to Mordor with the help of the shifty Gollum, the divided fellowship makes a stand against Sauron's new ally, Saruman, and his hordes of Isengard.

Save

Cancel

ngMessages: New in 1.3

Requires

- angular-messages
- ngMessages

ng-Messages

- Shows or hides messages
- Similar to a switch or case statement

ng-Message

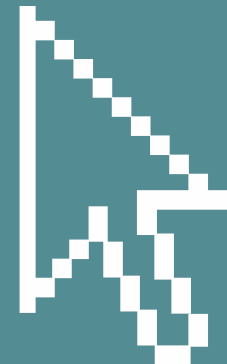
- Defines specific cases

```
<span class="help-block">  
  <span ng-messages="movieForm.inputMovieTitle.$error">  
    <span ng-message="required">Movie title is required</span>  
    <span ng-message="minlength">Must be at least 3 characters</span>  
    <span ng-message="maxlength">Cannot exceed 50 characters</span>  
  </span>  
</span>
```


DEMO:
HANDS ON LAB 5-3

Hands On Lab 5: Messages

1. Add the script tag for angular-messages
2. Add a dependency on ngMessages
3. Add validation messages
4. Run the application



Review – Forms & Validation



Building a Data Entry Form

Performing Validation

Setting Validation Styles

Displaying Validation Messages

Take Away

Expressive
HTML

Powerful
Data Binding

Modularity

Rule-based
Navigation

Easily Call
HTTP Service

Authoritative
Forms

Thank You!

- deborahk@insteptech.com
- jerryk@insteptech.com
- [@deborahkurata](#)
- <http://blogs.msmvps.com/deborahk>
- <https://github.com/DeborahK/angularu2015>